VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

POKROČILÁ OPTIMALIZACE TOKŮ V SÍTÍCH
ADVANCED OPTIMIZATION OF NETWORK FLOWS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Matouš Cabalka**
AUTHOR

VEDOUCÍ PRÁCE — **RNDr. Pavel Popela, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. Matouš Cabalka**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**RNDr. Pavel Popela, Ph.D.**
Akademický rok:	2017/18

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Pokročilá optimalizace toků v sítích

Stručná charakteristika problematiky úkolu:

Diplomant si prohloubí znalosti problematiky toků v sítích. Zúročí osvojené poznatky z teorie grafů, matematického programování, pravděpodobnosti a matematické statistiky. Zaměří se na pokročilé modely matematického programování a zejména na problematiku jejich efektivní modifikace a implementace. Diplomant naváže na své znalosti optimalizačních modelů v logistice a prohloubí si je ve vztahu k obecným principům modelování. Prostuduje vlastnosti vybraných úloh, naváže výběrem a modifikací vhodných algoritmů. Testovací výpočty bude realizovat s využitím reálných dat.

Cíle diplomové práce:

1. Teoretická část práce bude věnována vybraným poznatkům z teorie grafů a matematického programování.
2. Pozornost bude soustředěna na návaznosti zmíněných pojmů na vybrané úlohy z oblasti logistiku.
3. Diplomant se zaměří na tvorbu a řešení modelů rozhodování vhodných pro stochastické síťové problémy
4. Důraz bude kladen na jejich implementaci a na testovací výpočty.

Souhrnným cílem práce je přispět k řešení pokročilých síťových úloh s aplikacemi v logistice. Předpokládá se návaznost na problémy řešené v rámci NETME+ a spolupráce ústavu matematiky FSI VUT v Brně se specialisty v oblasti logistiky z norské Molde University College (prof. Kjetil Kare Haugen, prof. Asmund Olstad).

Seznam doporučené literatury:

BIRGE, John R. and François LOUVEAUX. Introduction to Stochastic Programming. Springer Verlag, 1997. ISBN: 978-1-4614-0236-7.

WOLSEY, Laurence A. Integer programming. New York: John Wiley & Sons, 1998. ISBN 978-0-4-1-28366-9.

GHIANI, Gianpaolo, Gilbert LAPORTE and Roberto MUSMANNO. Introduction to logistics systems planning and control. Hoboken, NJ, USA: J. Wiley, 2004. ISBN 0-470-84917-7.

NASH, Stephen and Ariela SOFER. Linear and nonlinear programming. McGraw-Hill, 1995. ISBN 978-0-89871-661-0.

WALLACE, Stein W. and Alan KING. Modeling with Stochastic Programming. Springer Verlag, 2012. ISBN 978-0-387-87816-4.

KALL, Peter and Stein W. WALLACE. Stochastic Programming. New York: John Wiley & Sons, 1993. ISBN 978-0471951582.

RUSZCZYNSKI, Andrzej et al. Handbooks in Operations Research and Management Science, vol. 10: Stochastic Programming. Amsterdam: Elsevier, 2003. ISBN 978-0-444-50854-6.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2017/18

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
Diplomová práce se zabývá optimalizačními modely v logistice s důrazem na úlohu napadení sítě. Po stručném úvodu následují dvě přehledové kapitoly věnované teorii grafů a matematickému programování. V kapitole pojmenované Základní pojmy z teorie grafů jsou uvedeny důležité pojmy, které úzce souvisí s úlohami napadání sítí. V návaznosti na zmíněné pojmy jsou uvedeny některé nezbytné věty, které se využívají při řešení problémů. V navazující kapitole pojmenované Úvod do matematického programování jsou nejprve uvedeny pojmy z lineárního programování. Pojmy jsou vybrány vzhledem k navazující úloze o maximálním toku v síti a k odvozené duální úloze. Následují základní pojmy stochastické optimalizace. V páté kapitole se věnujeme deterministickým modelům napadení sítě. Následuje kapitola věnovaná stochastickému napadání sítě. Každý model je zároveň implementován ve vlastním programu napsaném v programovacím jazyce GAMS, kódy jsou přiloženy.

Summary
The master's thesis focuses on the optimization models in logistics with emphasis on the network interdiction problem. The brief introduction is followed by two overview chapters - graph theory and mathematical programming. Important definitions strongly related to network interdiction problems are introduced in the chapter named Basic concepts of graph theory. Necessary theorems used for solving problems are following the definitions. Next chapter named Introduction to mathematical programming firstly contains concepts from linear programming. Definitions and theorems are chosen with respect to the following maximum flow problem and the derived dual problem. Concepts of stochastic optimization follow. In the fifth chapter, we discuss deterministic models of the network interdiction. Stochastic models of the network interdiction follow in the next chapter. All models are implemented in programmes written in the programming language GAMS, the codes are attached.

Klíčová slova
Matematické programování, lineární programování, toky v sítích, úloha o maximálním toku, duální úloha, úloha o minimálním řezu, úloha napadení sítě, dvoustupňové programování, stochastické programování, GAMS

Keywords
Mathematical programming, linear programming, network flows, the maximum flow problem, dual problem, the minimum cut problem, network interdiction problem, two-stage programming, stochastic programming, GAMS

CABALKA, M. *Pokročilá optimalizace toků v sítích.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 99 s. Vedoucí diplomové práce RNDr. Pavel Popela, PhD.

Prohlašuji, že jsem svou diplomovou práci na téma Pokročilá optimalizace toků v sítích vypracoval samostatně pod vedením vedoucího diplomové práce RNDr. Pavla Popely, PhD. s použitím odborné literatury a dalších zdrojů, které jsou uvedeny v seznamu literatury na konci práce.

Bc. Matouš Cabalka

Rád bych poděkoval vedoucímu diplomové práce panu RNDr. Pavlu Popelovi, PhD. za odborné vedení, konzultace, trpělivost a četné připomínky při vedení mé práce. Velký dík patří také Bc. Filipu Korbelovi za četné rozhovory na témata související s mojí diplomovou práci. Největší poděkování ale pochopitelně patří rodině, která mne bezmezně podporovala během celé doby studia doma i v zahraničí.

Bc. Matouš Cabalka

Obsah

1. Úvod

Cílem této diplomové práce je studium modelů a metod řešení úloh optimalizace toků v sítích s důrazem na úlohy napadání sítě (angl. *network interdiction problems*, [37]). Napadáním sítě se rozumí přerušování nebo kontrola toku v uzlech, hranách nebo kombinace obojího.

Z pohledu rozsáhlé historie matematiky se jedná o relativně mladé téma, neboť úlohám tohoto typu začala být větší pozornost věnovaná poprvé v průběhu války ve Vietnamu v 60. a 70. letech dvacátého století, viz [31]. Přestože tato problematika čerpá z teoretických základů pocházejících z předchozí dekády, zásadní zlom přichází v roce 1962, kdy Ford a Fulkerson formulovali svoji známou větu z teorie grafů o maximálním toku a minimálním řezu, [10]. Věta dává do souvislosti, velice stručně řečeno, maximální tok v síti s kapacitou minimálního řezu. S použitím algoritmu pro hledání maximálního toku lze určit, které hrany náleží do minimálního řezu a které je tedy vzhledem k záměru minimalizace maximálního toku v síti vhodné přerušit.

Na práci Forda a Fulkersona navazuje v roce 1970 Wollmer, který sestavil dva heuristické algoritmy, viz [35], s jejichž pomocí lze určit nejdůležitější hranu v komunikační síti, jejíž zničení podstatně zvýší cenu používání sítě. V roce 1971 Helmbold navazuje na Wollmera a využívá k modelování dané problematiky dynamické programování, viz [14]. V průběhu následujících let se samozřejmě rozrůstá nejen množství způsobů jak problémy modelovat, ale i množství možných aplikací.

Zatímco první aplikaci představovalo napadání komunikačních sítí ve Vietnamu během války, v 90. letech dvacátého století na tuto aplikaci plynule navazuje snaha SOUTHCOMu (složka americké armády spadající pod americké ministerstvo obrany) zabránit pašování drog z Jižní Ameriky, viz [37]. S rozvojem moderní techniky a dostupnosti různých zbraňových systémů se o podobné modely začala kolem roku 2000 zajímat i americká armáda a to především kvůli riziku pašování jaderného materiálu, zejména v oblastech postsovětských zemí, Ruska a střední Asie, viz [25].

Na tuto problematiku není nutné pohlížet jako na snahu někomu uškodit. Naším záměrem samozřejmě není věnování se polovojenským nebo vojenským aplikacím. Pokud se přesuneme „na druhou stranu barikády“ a staneme se ne tím, kdo síť kontroluje, ale tím, kdo síť využívá, objeví se překvapivě velké množství potenciálních aplikací. Ve snaze uvést další aplikace, využijeme [33], kde je značné množství nevojenských aplikací zmíněno.

Jednou z možností je pohled na síť z hlediska její spolehlivosti. Představme si rozvodnou elektrickou síť. Společnost distribuující elektřinu může zajímat, co se stane, když přijde přírodní katastrofa a některé komponenty její rozvodné sítě budou poškozeny. Kolik lidí pak bude bez proudu? Jak drahé a náročné budou opravy? Jaký dopad bude mít pád vedení vysokého napětí a jaký dopad bude mít zničení trafostanice za městem?

Obdobné aplikace samozřejmě vyvstávají i v oblasti zdravotní péče v případě přírodních katastrof. Nejen ohledně dostupnosti péče, ale pochopitelně i ohledně kapacit zdravotních zařízení. V době moderních technologii nesmíme opomenout ani různé komunikační sítě nejen z hlediska technologií, ale i jejich uživatelů.

Jako nejaktuálnější témata stojí za zmínku bezpečnostní rizika spojená nejen s terorismem, ale i migrací. Až doposud jsme uvažovali náhodné vlivy. Například, že netušíme, kde konkrétně v Jižní Americe se drogy produkují, nebo že nevíme, kde přírodní katastrofa

udeří. Uvažovat lze ale samozřejmě také možnost, že útočník je v jistém slova smyslu racionální, zná dokonale síť a ví, jak udeřit, aby maximalizoval svůj dopad.

Abychom se mohli problematice věnovat a sestavovat vhodné modely, je potřeba vyjít z teoretického základu. První stavební kámen představuje kapitola uvádějící základy teorie grafů nejen kvůli názornosti ilustrovaných pojmů, ale také kvůli snadno srozumitelné interpretaci. Navíc právě z teorie grafů vychází některé pro tuto práci podstatné věty, připomeňme dříve zmíněnou větu o maximálním toku a minimálním řezu. Na teorii grafů navazuje další stavební kámen v podobě kapitoly uvádějící do základů matematického programování. Kapitola je koncipována tak, aby byly nejprve uvedeny důležité pojmy související s následně uvedenými konstrukcemi a modely. Mezi tyto konstrukce počítáme metody stochastického programování a úlohy dvoustupňové stochastické optimalizace.

Na přehledové kapitoly navazují konkrétní deterministické modely napadení sítě, na nichž jsou provedeny testovací výpočty ve vlastní programech napsaných v programovacím jazyce GAMS. Součástí výpočtů je výstup z programu GAMS, kde jsou výpočty prováděny, a grafická interpretace výsledků ve formě obrázků znázorňujících modelovou síť po napadení útočníkem.

Na deterministické modely navazují modely stochastických reformulací, kde jsou nejprve náhodně přerušovány hrany pro tři scénáře na názorných příkladech, pak jsou modely testovány na rozsáhlejších datech pro více vnitřních uzlů a více scénářů. V závěru pak následuje dvoustupňová úloha, v níž povolíme opětovné zapnutí některých hran za cenu kompenzace. Vše je doplněno o výstupy z programu GAMS, obrázky a komentáře.

2. Základní pojmy z teorie grafů

V matematice může být někdy pro studenty obtížné si pod teoretickými pojmy něco představit, natož najít nějakou souvislost mezi pojmy navzájem. Zmíněný problém se ale netýká teorie grafů, která nachází četné aplikace v matematice, informačních technologiích, logistice nebo společenských vědách, například v [9], [26]. Teorie grafů nabízí efektivní a srozumitelný nástroj vhodný nejen k názorné reprezentaci některých problémů, ale i k jejich řešení.

Ve své práci uvádím důležité definice a vysvětlení, která dále využiji při sestavení a řešení síťových úloh. Cílem je uvést čtenáře do problematiky teorie grafů a seznámit ho se základními pojmy teorie grafů. Pojmy jsou vybírány vzhledem k navazujícím logistickým aplikacím z [9] a [26].

Definice 2.1. (Graf)
Graf G je uspořádaná dvojice (N, E), kde $N = \{n_1, n_2, ..., n_n\}$ je množina uzlů (nebo také vrcholů) grafu a $E = \{e_1, e_2, ..., e_m\}$ je množina hran spojujících všechny nebo některé z uzlů.

Definice 2.2. (Prostý neorientovaný graf)
Prostý neorientovaný graf je dvojice $G = (N, E)$, kde N je konečná množina uzlů a $E = \{\{x, y\} : x, y \in N \wedge x \neq y\}$ je konečná množina neorientovaných hran. Říkáme, že hrana $e = \{x, y\}$ je incidentní s uzly x, y, nebo že spojuje uzly x a y.

Definice 2.3. (Prostý orientovaný graf)
Prostý orientovaný graf je dvojice $G = (N, A)$, kde N je konečná množina uzlů a $A = \{(x, y) : x, y \in N \wedge x \neq y\}$ je konečná množina orientovaných hran.

(a) Prostý neorientovaný graf (b) Prostý orientovaný graf

Obrázek 2.1: Příklady prostých grafů.

Poznámka:
Podle výše uvedených definic existují dvě různá propojení hran, která je třeba rozlišovat, neorientované a orientované hrany, viz Obr. 2.1. Neorientovanou hranu zpravidla znázorňujeme jako úsečku, která reprezentuje obousměrné propojení uzlů, lze ji ale také znázornit jako úsečku s šipkami na obou koncích. Orientovanou hranu zakreslíme jako úsečku se šipkou na jednom konci. Reprezentujeme tak jednosměrný vztah, například jednosměrnou ulici. Množinu neorientovaných hran budeme značit jako E (z anglického „*edge*"), množinu orientovaných hran budeme značit A (z anglického „*arc*"). Některé dále definované pojmy budeme zpravidla pro snazší představu ilustrovat obrázky s neorientovanými, jiné s orientovanými hranami. Důležitou roli zde bude hrát označení jednotlivých typů grafů odkazující na příslušný typ hran.

Definice 2.4. (Neorientovaný graf (multigraf))
Neorientovaný graf G je trojice (N, E, ϵ), kde N je konečná množina uzlů, E je konečná množina neorientovaných hran a ϵ je zobrazení, které každé hraně přiřazuje dvojici rozdílných uzlů tak, že $\epsilon : E \to \{\{x, y\} : x, y \in N \wedge x \neq y\}$. Každá dvojice uzlů může incidovat s více hranami.

Definice 2.5. (Orientovaný graf (multigraf))
Orientovaný graf G je trojice (N, A, ϵ), kde N je konečná množina uzlů, A je konečná množina hran a ϵ je zobrazení přiřazující každé hraně uspořádanou dvojici rozdílných uzlů tak, že $\epsilon : A \to \{(x, y) : x, y \in N \wedge x \neq y\}$. Mezi každou uspořádanou dvojicí uzlů může být více orientovaných hran.

Poznámka:
Zásadní rozdíl mezi prostým neorientovaným grafem a neorientovaným multigrafem je ten, že v prvém může mezi dvěma určitými uzly být nejvýše jedna hrana. Ve druhém případě může být hran více, viz Obr. 2.2.

(a) Neorientovaný multigraf

(b) Orientovaný multigraf

Obrázek 2.2: Příklady multigrafů.

Poznámka:
V literatuře lze nalézt řadu dalších definic různých typů grafů, viz [9], [26], [28]. Mezi nejznámější námi nedefinované grafy patří grafy se *smyčkami*, což jsou hrany začínající a končící v témže uzlu. Tyto grafy ale neuvádíme záměrně, protože z hlediska síťových aplikací, kterým se budeme věnovat, nepřináší žádnou změnu řešení.

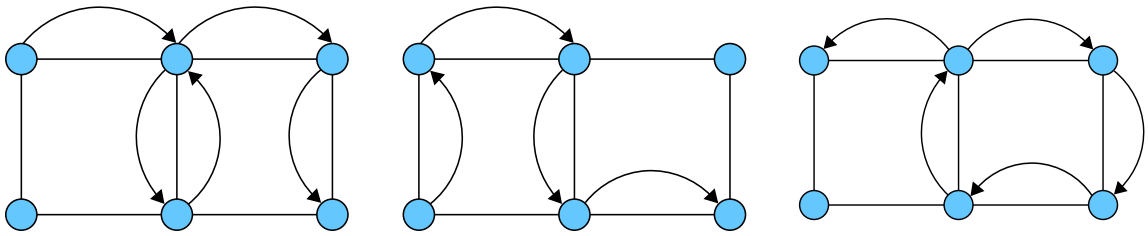
Definice 2.6. (Podgraf)
Podgrafem grafu G s množinou uzlů N(G) rozumíme libovolný graf H na podmnožině uzlů $N(H) \subseteq N(G)$, který má za hrany libovolnou podmnožinu hran grafu G majících oba uzly v $N(H)$. Píšeme $H \subseteq G$.

Definice 2.7. (Sled)
Sled délky k v grafu (N, A, ϵ) z uzlu x_0 do uzlu x_k je konečná posloupnost tvaru $x_0, a_1, x_1, a_2, x_2, ..., x_{k-1}, a_k, x_k$, kde $k \geq 0$, ve které se vždy střídají uzly a hrany ($x_i \in N, a_i \in A$ pro $\forall i = 0, ..., k$) a pro každé $j = 1, ..., k$ platí, že $\epsilon(a_j) = (x_{j-1}, x_j)$, tedy říkáme, že hrana a_j propojuje uzel x_{j-1} s uzlem x_j, viz Obr. 2.3a.

Definice 2.8. (Cesta)
Cesta v grafu je sled, v němž se žádné uzly neopakují, viz Obr. 2.3b.

Definice 2.9. (Tah)
Tah je sled v grafu, v němž se neopakují žádné hrany, viz Obr. 2.3c.

Poznámka:
V grafech existují dva typy cest a tahů, otevřené a uzavřené. O uzavřených cestách a tazích hovoříme v případě, když končí v uzlech, v nichž začaly. V opačném případě mluvíme o otevřených cestách a tazích.



(a) Sled (b) Cesta (c) Tah

Obrázek 2.3: Příklady sledů.

Definice 2.10. (Slabě souvislý graf)
Řekneme, že orientovaný graf G je slabě souvislý, pokud jsou všechny jeho uzly propojeny nehledě na orientaci.

Definice 2.11. (Silně souvislý graf)
Řekneme, že orientovaný graf G je silně souvislý, jestliže pro každé dva různé uzly grafu $x, y \in N(G)$ existuje v G orientovaná cesta z x do y.

Poznámka:
V případě neorientovaného grafu pojmy slabé a silné souvislosti grafu splývají, protože neorientované hrany představují „obousměrnou ulici“. Pokud je neorientovaný graf slabě souvislý, lze tedy najít cestu z kteréhokoli do kteréhokoli uzlu. V takovém případě stačí říct jednoduše, že neorientovaný graf je souvislý.

Definice 2.12. (Komponenta grafu)
Komponentou grafu G nazýváme každý podgraf G' grafu G, který je souvislý a maximální, tj. je-li $G'' \neq G'$ jiný podgraf grafu G takový, že $G' \subseteq G''$, pak již G'' není souvislý.

Definice 2.13. (Úplný graf)
Graf $G = (N, E)$ se nazývá úplný, jestliže pro každé dva uzly x, $y \in N$, $x \neq y$ existuje alespoň jedna hrana e, která je spojuje. Má-li n uzlů, značíme jej K_n.

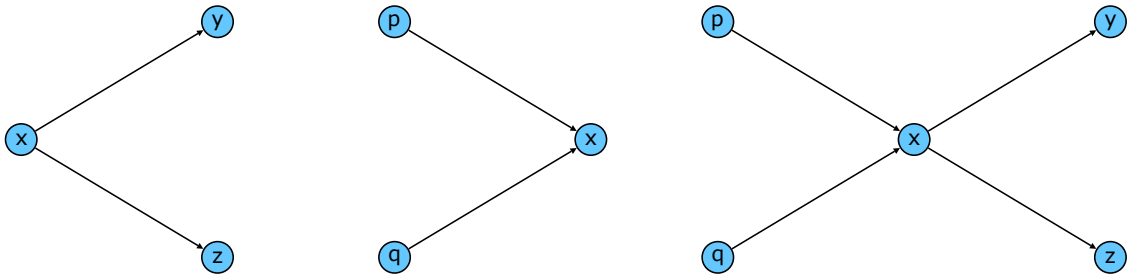
Poznámka:
Orientovaný graf $G = (N, A)$ je úplný, pokud množina hran A obsahuje všechny uspořádané dvojice různých uzlů z množiny N, tedy existuje orientovaná hrana vedoucí mezi každými dvěma různými uzly v obou směrech.

Definice 2.14. (Most)
Řekneme, že hrana e v grafu $G = (N, E, \epsilon)$ je most právě tehdy, když počet komponent grafu vzroste jejím odstraněním.

Poznámka:
Hrana a v orientovaném grafu $G = (N, A)$ je silným mostem, jestliže jejím odstraněním vzroste počet silně souvislých komponent grafu G.

Definice 2.15. (Množiny hran a uzlů v orientovaných grafech)
Mějme orientovaný graf $G = (N, A, \epsilon)$ a nechť $x, y \in N$ jsou jeho libovolné uzly a $M \subseteq N$ nechť je libovolná množina jeho uzlů. $P(a)$ nechť značí počáteční uzel hrany a a $K(a)$ nechť značí koncový uzel hrany a, tj. hrana $\epsilon(a) = (x, y)$ má $P(a) = x$ a $K(a) = y$. Pak zavedeme následující pojmy a značení:

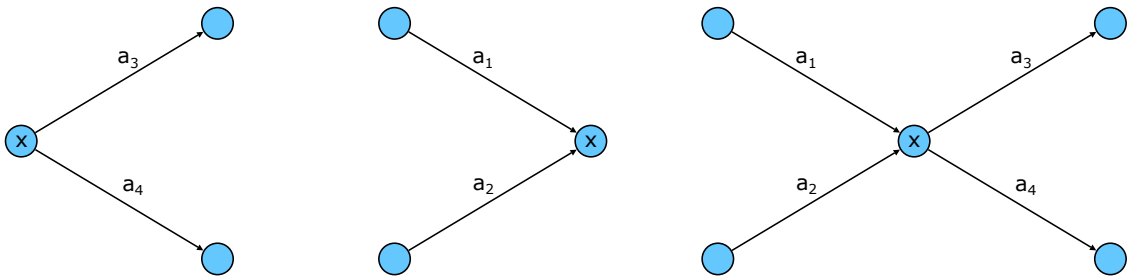
- $N_G^+(x) = \{y \in N | (x, y) \in \epsilon(A)\}$ je množina následníků uzlu x, tj. množina uzlů, do nichž vede hrana z x, viz Obr. 2.4a.
- $N_G^-(x) = \{y \in N | (y, x) \in \epsilon(A)\}$ je množina předchůdců uzlu x, tj. množina uzlů, z nichž vede hrana do x, viz Obr. 2.4b.
- $N_G(x) = N_G^+(x) \cup N_G^-(x)$ je množina sousedů uzlu x, tj. množina uzlů spojených hranou s uzlem x, viz Obr. 2.4c.
- $N_G(M) = \bigcup_{x \in M} N_G(x)$, tj. množina uzlů spojených hranou s některým uzlem z M.



(a) $N_G^+(x) = \{y, z\}$ (b) $N_G^-(x) = \{p, q\}$ (c) $N_G(x) = \{p, q, y, z\}$

Obrázek 2.4: Ilustrace různých množin sousedů uzlu x.

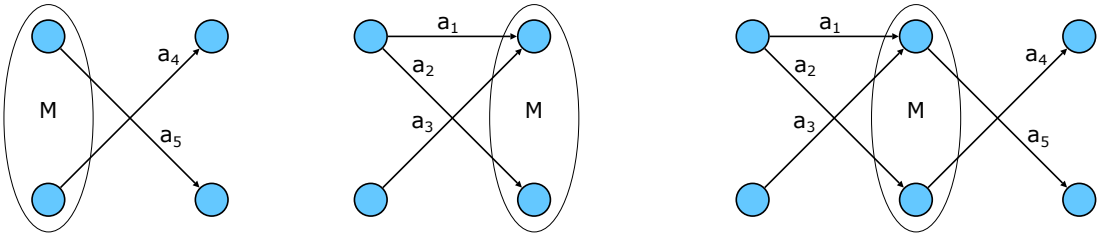
- $A_G^+(x) = \{a \in A | P(a) = x\}$ je výstupní okolí uzlu x, tj. množina hran s počátečním uzlem x, viz Obr. 2.5a.
- $A_G^-(x) = \{a \in A | K(a) = x\}$ je vstupní okolí uzlu x, tj. množina hran s koncovým uzlem x, viz Obr. 2.5b.
- $A_G(x) = A_G^+(x) \cup A_G^-(x)$ je okolí uzlu x, tj. množina hran majících uzel x za počáteční, nebo cílový uzel, viz Obr. 2.5c.



(a) $A_G^+(x) = \{a_3, a_4\}$ (b) $A_G^-(x) = \{a_1, a_2\}$ (c) $A_G(x) = \{a_1, a_2, a_3, a_4\}$

Obrázek 2.5: Ilustrace různých okolí uzlů x.

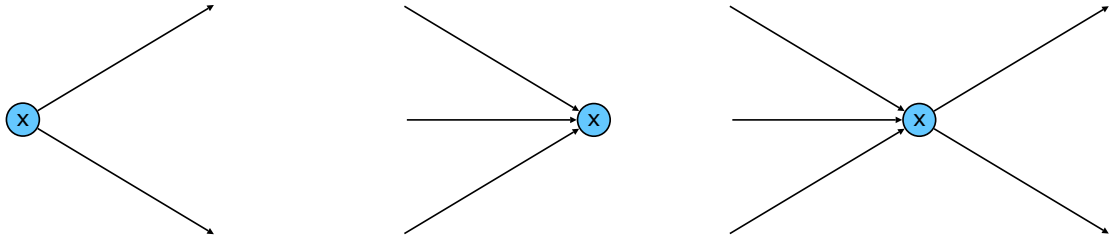
- $W_G^+(M) = \{a \in A | P(a) \in M \wedge K(a) \notin M\}$, tj. množina všech hran, jejichž počáteční uzel leží v M a koncový uzel neleží v M, viz Obr. 2.6a.
- $W_G^-(M) = \{a \in A | P(a) \notin M \wedge K(a) \in M\}$, tj. množina všech hran, jejichž počáteční uzel neleží v M a koncový uzel leží v M, viz Obr. 2.6b.
- $W_G(M) = W_G^+(M) \cup W_G^-(M)$ je množina všech hran, jejichž jeden (libovolný) uzel leží v množině M a druhý v množině M neleží. Takovou množinu nazýváme řez určený množinou M, viz Obr. 2.6c.



(a) $W_G^+(M) = \{a_4, a_5\}$ (b) $W_G^-(M) = \{a_1, a_2, a_3\}$ (c) $W_G(M) = \{a_1, ..., a_5\}$

Obrázek 2.6: Ilustrace pojmů souvisejících s řezy množiny M.

- $d_G^+(x) = |A^+(x)|$ je výstupní stupeň uzlu x, tj. počet hran vystupujících z uzlu x, viz Obr. 2.7a.
- $d_G^-(x) = |A^-(x)|$ je vstupní stupeň uzlu x, tj. počet hran vstupujících do uzlu x, viz Obr. 2.7b.
- $d_G(x) = d_G^+(x) \cup d_G^-(x)$ je stupeň uzlu x, tj. počet hran, pro které je x počátečním nebo cílovým uzlem. Případné smyčky se počítají dvakrát, viz Obr. 2.7c.



(a) $d_G^+(x) = 2$ (b) $d_G^-(x) = 3$ (c) $d_G(x) = 5$

Obrázek 2.7: Ilustrace různých stupňů uzlu x.

Poznámka:
Zpravidla bude z kontextu jasné, který graf je předmětem našeho zájmu a proto budeme záměrně vynechávat index $_G$. Místo $N_G(x)$ budeme psát $N(x)$. Navíc zjednodušíme značení ve smyslu zobrazení $\epsilon : A \to N^2$, tedy místo $\epsilon(a_{ij}) = (n_i, n_j)$ budeme psát $a_{ij} = (n_i, n_j)$ a místo $G = (N, A, \epsilon)$ budeme psát $G = (N, A)$.

2.1. Reprezentace grafů

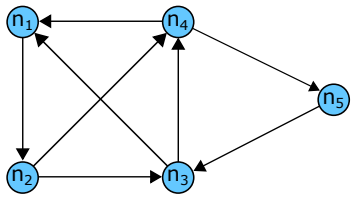
Definice 2.16. (Matice sousednosti)
Uvažujme orientovaný graf $G = (N, A)$. Potom řekneme, že matice $\mathbf{M} = (m_{ij})$ daná vztahem

$$m_{ij} = \begin{cases} 1 & \text{jestliže hrana } (n_i, n_j) \in A, \\ 0 & \text{jestliže hrana } (n_i, n_j) \notin A, \end{cases}$$

je matice sousednosti grafu G, kde n_i, $n_j \in N$. Jestliže jsou hrany grafu G ohodnoceny, pak místo jedniček vkládáme do matice délku hrany a_{ij} a říkáme jí matice vzdáleností, nebo distanční matice.

Příklad 2.17. Příklad matice sousednosti $\mathbf{M}$ pro orientovaný graf uvedený na obrázku, kde $N = \{n_1, n_2, n_3, n_4, n_5\}$ a $A = \{a_{12}, a_{23}, a_{24}, a_{31}, a_{34}, a_{41}, a_{45}, a_{53}\}$.

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



Definice 2.18. (Symetrický graf)
Graf G je symetrický, jestliže má symetrickou matici sousednosti.

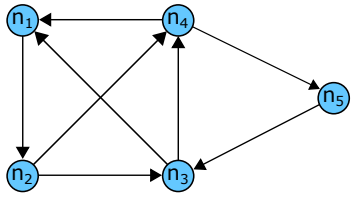
Definice 2.19. (Matice incidence)
Nechť $G = (N, A)$ je orientovaný graf bez smyček. Jestliže libovolně pevně zvolíme nejen pořadí uzlů $\{n_1, ..., n_n\}$, ale i pořadí hran $\{a_1, a_2, ..., a_m\}$, můžeme grafu G přiřadit matici incidence $\mathbf{M} = (m_{ij})$ typu (n, m) takto

$$m_{ij} = \begin{cases} -1 & \text{jestliže } n_i \text{ je koncový uzel hrany } a_j, \\ 1 & \text{jestliže } n_i \text{ je počáteční uzel hrany } a_j, \\ 0 & \text{jinak.} \end{cases}$$

Poznámka:
Abychom ukázali rozdílnost zápisu grafu pomocí matice sousednosti a matice incidence, využijeme stejný graf jako v příkladu 2.17.

Příklad 2.20. Příklad matice incidence $\mathbf{M}$ pro orientovaný graf uvedený na obrázku, kde $N = \{n_1, n_2, n_3, n_4, n_5\}$ a $A = \{a_{12}, a_{23}, a_{24}, a_{31}, a_{34}, a_{41}, a_{45}, a_{53}\}$.

$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & -1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$



2.2. Toky v sítích

V této podkapitole budeme pokračovat ve využívání značení převzatého z [9], odkud také pochází následující definice a věty.

Definice 2.21. (Síť)
Síť je čtveřice $S = (G, s, t, u)$, kde $G = (N, A)$ je orientovaný graf, s, $t \in N$ dva různé uzly grafu G takové, že do s nevstupují žádné hrany a z uzlu t žádné hrany nevystupují, u je zobrazení $u : A \to \mathbb{R}$. Funkce u přiřazuje každé orientované hraně $a \in A$ grafu G kladné reálné číslo $u(a)$.

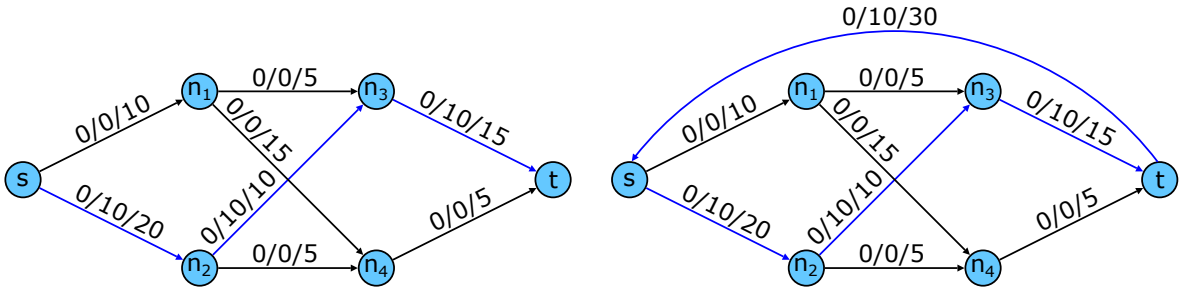
Poznámka:
Uzel s se nazývá zdroj, nebo také zdrojový uzel (z anglického „*source*"), pokud pro něj platí, že neexistuje hrana $(n_i, s), \forall n_i \in N$. Uzel t se nazývá spotřebič nebo také cílový uzel (z anglického „*terminus*"), pokud pro něj platí, že neexistuje hrana $(t, n_i), \forall n_i \in N$. Číslo $u(a)$ se pak nazývá kapacita hrany a.

Definice 2.22. (Tok v síti)
Tok v síti $S = (G, s, t, u)$ je takové ohodnocení hran grafu $G = (N, A)$ reálnými čísly (tj. zobrazení $x : A \to \mathbb{R}$), které splňuje následující podmínky:

1. *Kapacitní omezení*: Tok v žádné hraně nemůže převýšit její kapacitu nebo být záporný, tj.
$$\forall a \in A : 0 \leq x(a) \leq u(a)$$
2. *Kirchhoffův zákon zachování toku*: Přítok do každého vnitřního uzlu n_i je roven odtoku z uzlu n_i, tj.
$$\forall n_i \in N \setminus \{s, t\} : \sum_{a \in A_G^-(n_i)} x(a) = \sum_{a \in A_G^+(n_i)} x(a)$$

Pokud Kirchhoffův zákon platí pro všechny uzly včetně počátečního a cílového, místo toku mluvíme o cirkulaci. Jestliže je v síti více počátečních uzlů, přidáme do sítě takzvaný super-zdrojový uzel S. Označíme $S' = \{s_1, ..., s_n\}$ množinu všech počátečních uzlů v síti. Pak platí, že existují hrany $(S, s_i), \forall s_i \in S'$ s kapacitami dostatečně velkými, aby nelimitovali tok v síti. Dostatečně velkou kapacitou může být nekonečná kapacita, nebo kapacita rovna součtu všech kapacit hran, které vystupují ze všech počátečních uzlů. Analogicky bychom postupovali v případě, pokud by síť obsahovala více cílových uzlů. Vnitřními uzly myslíme všechny uzly kromě počátečních a cílových, tedy v síti lze množinu uzlů N rozdělit na množinu počátečních uzlů S', množinu vnitřních uzlů N' a množinu cílových uzlů T', kde každý uzel z množiny N náleží do právě jedné z těchto tří množin.

Poznámka:
Rozdíl mezi tokem a cirkulací je přirozený. Pokud Kirchhofův zákon platí pouze pro vnitřní uzly, znamená to, že v počátečním uzlu tok vzniká a v cílovém uzlu tok zaniká. Je to, jako kdybychom posílali autobus s cestujícími z jedné konečné stanice do druhé, zatímco v případě cirkulace bychom mohli uvažovat uzavřené potrubí v němž stále koluje tentýž objem kapaliny. Pojmy ilustrujeme na následujícím obrázku, kde první číslo na hraně reprezentuje dolní mez toku hranou $l(a)$, druhé číslo tok hranou $x(a)$ a třetí číslo kapacitu hrany $u(a)$.



(a) Tok v síti (b) Cirkulace

Obrázek 2.8: Ukázka toku a cirkulace.

Definice 2.23. (Velikost toku od zdroje ke spotřebiči)
Mějme síť $S = (G, s, t, u)$. Velikost toku od zdroje ke spotřebiči definujeme jako množství toku, které vzniká ve zdroji, tj. jako rozdíl

$$F(x) = \sum_{a \in A^+(s)} x(a) \quad - \sum_{a \in A^-(s)} x(a).$$

Velikost toku od zdroje ke spotřebiči by bylo možné obdobně definovat jako množství toku, které zaniká ve spotřebiči, tedy pokud bychom zaměnili pořadí sum a uvažovali okolí cílového uzlu t.

Definice 2.24. (Řez)
Řez (P, Q) je taková podmnožina hran, která rozdělí množinu uzlů N na dvě podmnožiny P a $Q = N \setminus P$. Skládá se z takových hran, jejichž jeden libovolný bod leží v množině P a druhý leží v množině Q. Hrany (p, q), kde $p \in P$ a $q \in Q$, nazveme dopředné hrany řezu (P, Q) a jejich kapacitu označíme $C(P, Q)$. Hrany (q, p) pak nazveme zpětné hrany řezu (P, Q) a jejich kapacitu označíme jako $C(Q, P)$. Pokud máme síť $S = (G, s, t, u)$, množina uzlů P obsahuje počáteční uzel sítě s a množina Q obsahuje cílový uzel sítě t, pak řez nazýváme $s - t$ řez s kapacitou $C(P, Q)$. Takových $s - t$ řezů v síti může být více a mohou mít různé kapacity. Jejich důležitou vlastností je, že oddělí množinu uzlů obsahující počáteční uzel s od množiny uzlů obsahující cílový uzel t. Abychom zdůraznili vazbu na řez, budeme značit kapacitu s-t řezu $C(S, T)$, $s \in S$, $t \in T$, $S \subset N$, $T = N \setminus S$.

Definice 2.25. (Velikost toku přes řez)
Velikost toku přes řez $W(M)$ určený množinou uzlů M definujeme jako množství toku, které hranami řezu $W(M)$ teče z množiny uzlů M do $N \setminus M$ zmenšené o množství, které se hranami řezu $W(M)$ zpět do množiny M vrací z $N \setminus M$, tj.

$$F_M(x) = \sum_{a \in W^+(M)} x(a) \quad - \sum_{a \in W^-(M)} x(a).$$

Za zmínku stojí, že velikost toku od zdroje ke spotřebiči jsme definovali v definici 2.23 jako tok přes speciální řez určený množinou $M = \{s\}$.

Věta 2.26.
Nechť x je libovolný tok z počátečního uzlu s do cílového uzlu t a nechť $W(M)$ je libovolný řez, který odděluje počáteční a cílový uzel, tj. $s \in M$ a $t \in N \setminus M$. Pak platí, že $F_M(x) = F(x)$, tj. velikost $F(x)$ toku od zdroje ke spotřebiči je rovna velikosti $F_M(x)$ toku přes řez $W(M)$.

Poznámka:
Větu ponecháme bez důkazu. Intuitivně lze na větu nahlížet jako na zákon zachování hmoty. Návod na důkaz je uveden v [9].

Poznámka:
Tok v síti lze převést na cirkulaci přidáním pomocné návratové hrany z cílového uzlu t do počátečního uzlu s s dostatečně velkou kapacitou, kterou může být například součet kapacit všech hran, které vycházejí z počátečního uzlu s. Vzhledem ke kapacitnímu omezení a platnosti Kirchhoffova zákona lze pak úlohu o maximálním toku v síti, které se budeme věnovat v následující kapitole, řešit zároveň jako maximalizaci toku v síti přes návratovou hranu.

Definice 2.27. (Kapacita řezu)
Kapacitu $C(M)$ řezu $W(M)$ definujeme takto

$$C(M) = \sum_{a \in W^+(M)} u(a) \quad - \sum_{a \in W^-(M)} l(a),$$

kde $u(a)$ je kapacita hrany (horní mez toku hranou) a $l(a) \geq 0$ je dolní mez toku hranou a, tedy $\forall a \in A : l(a) \leq x(a) \leq u(a)$. Význam kapacity řezu spočívá v tom, že žádný tok z počátečního do cílového uzlu nemůže být větší než kapacita kteréhokoli řezu, který odděluje počáteční a cílový uzel, což formulujeme následující větou.

Věta 2.28.
Pro kapacitu libovolného řezu $W(M)$, který odděluje zdroj a spotřebič a libovolný přípustný tok x platí vztah $F(x) \leq C(M)$.

Důkaz:
Velikost toku mezi zdrojem a spotřebičem $F(x)$ se rovná velikosti toku $F_M(x)$ přes řez $W(M)$, viz věta 2.26. Platnost nerovnosti $F_M(x) \leq C(M)$ plyne z faktu, že tok je přípustný, a tedy $F(x) \leq C(M)$.

Věta 2.29.
Uvažujme síť s dolními omezeními toku l a horními omezeními toku u. Přípustná cirkulace existuje právě tehdy, když má každý řez $W(M)$ nezápornou kapacitu, tj.

$$C(M) = \sum_{a \in W^+(M)} u(a) \quad - \sum_{a \in W^-(M)} l(a) \geq 0.$$

Pokud má každý řez $W(M)$ nezápornou kapacitu $C(M)$, pak lze realizovat tok v síti, jehož velikost je menší nebo rovna nejmenší kapacitě řezu. Jestliže lze v síti realizovat tok, pak lze v síti realizovat cirkulaci přidáním návratové hrany z cílového uzlu t do počátečního uzlu s. Pokud zvolíme kapacitu návratové hrany dostatečně velkou, například rovnou kapacitě nejmenšího řezu, pak v případě realizace cirkulace je tok návratovou hranou roven toku v síti. Jak jsme uvedli v definici 2.22, Kirchhoffův zákon v případě cirkulace pak platí pro všechny uzly, včetně počátečního uzlu s a cílového uzlu t.

Důkaz:
Jestliže existuje přípustná cirkulace x, pak pro libovolný řez platí:

$$C(M) = \sum_{a \in W^+(M)} u(a) \quad - \sum_{a \in W^-(M)} l(a) \quad \geq \sum_{a \in W^+(M)} x(a) \quad - \sum_{a \in W^-(M)} x(a) = 0.$$

Rovnost vlevo plyne z definice 2.27. Rovnost vpravo je analogií Kirchhoffova zákona pro uzly z definice 2.22. V síti by velikost toku přes řez z definice 2.25 byla kladná, ale zde díky cirkulaci tok stále koluje v síti skrz návratovou hranu. Množství toku, které z řezu vyteče, se do něj musí také vrátit. Nerovnost pak platí, protože množství jakéhokoli toku nemůže být větší než kapacita libovolného řezu. Pokud přípustná cirkulace neexistuje, existence řezu se zápornou kapacitou vyplyne z algoritmu pro hledání přípustné cirkulace, viz [9]. To ale není náš případ, protože my uvažujeme $l(a) \geq 0, \forall a \in A$.

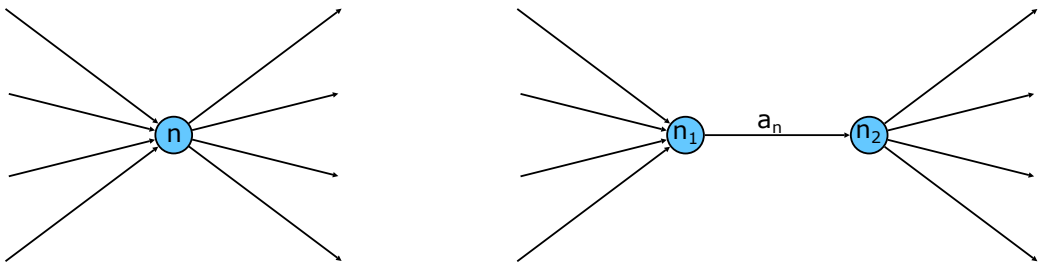
Poznámka:
V dalších částech textu budeme často používat termín „minimální řez“. Každý může vnímat minimální řez jiným způsobem, například jako minimální počet hran nutný k přerušení toku v síti od zdroje ke spotřebiči. V našem případě budeme v kapitole 5 přerušovat hrany, zpravidla ne všechny. „Minimální řez“ tedy bude takový řez, který po přerušení určitého počtu hran bude mít nejmenší kapacitu.

Poznámka:
Jestliže nastane situace, kdy bude v síti několik počátečních a několik cílových uzlů, lze pro zjednodušení převést úlohu na klasickou úlohu s jedním počátečním a jedním cílovým uzlem.

K síti přidáme dva uzly, umělý počáteční uzel S a umělý cílový uzel T. Každý původní počáteční uzel spojíme novými hranami s umělým počátečním uzlem. Každý původní cílový uzel spojíme novými hranami s umělým cílovým uzlem. Kapacity nových hran volíme tak, abychom nijak neomezovali toky z počátečních nebo do cílových uzlů, pokud to není náš záměr.

Takové zjednodušení využijeme v následujícím textu, například při formulaci modelu v podkapitole 5.3.

Poznámka:
V reálném světě existují četné úlohy, kde se neomezuje pouze tok v hranách, ale také tok přes uzel čili propustnost uzlu. Situaci lze řešit dvěma způsoby. Prvním způsobem je rozdělení uzlu n, u něhož chceme omezit propustnost, na dva uzly n_1, n_2 a přidání pomocné hrany a_n mezi dva nově vzniklé uzly, viz obrázek 2.9.



Obrázek 2.9: Převedení omezené propustnosti uzlu na omezenou propustnost hrany.

Pro naše účely je postačující uvažovat pouze sítě, v nichž jsou přípustné pouze nezáporné toky. Všechny hrany vstupující do uzlu n budou nyní vstupovat do uzlu n_1. Všechny hrany vystupující z uzlu n budou přirozeně vystupovat z uzlu n_2 a pomocná hrana a_n povede z uzlu n_1 do uzlu n_2. Tok protékající skrz uzel n lze nyní omezit pomocí toku protékajícího skrz hranu a_n. Tento způsob uvádíme z historického hlediska a pro úplnost, abychom naznačili, jak se k této problematice přistupovalo dříve. Druhým způsobem je

vynásobení kapacit vstupních hran takovou konstantou, aby byla nejvyšší propustnost uzlu rovna součtu kapacit vstupních hran.

Definice 2.30. (Minimální řez)
Uvažujme síť $S = (G, s, t, u)$. Minimální s-t řez je takový, jehož kapacita $C(S, T)$ je nejmenší.

Definice 2.31. (Maximální tok)
Uvažujme toky x a x' v síti $S = (G, s, t, u)$. Maximální tok x je takový, jehož velikost $F(x) \geq F(x')$, kde x' představuje libovolný tok z počátečního uzlu s do cílového uzlu t.

Věta 2.32. (Ověření optimálnosti toku a řezu)
Předpokládejme, že x je přípustný tok (tj. tok, jehož velikost je menší nebo rovna kapacitě nejmenšího řezu) z počátečního do cílového uzlu a že $W(M)$ je řez oddělující počáteční a cílový uzel. Jestliže je velikost $F(x)$ toku x rovna kapacitě $C(M)$ řezu $W(M)$, tj. $F(x) = C(M)$, pak tok x je maximálním tokem z počátečního uzlu s do cílového uzlu t a řez $W(M)$ je minimálním řezem.

Důkaz:
Velikost žádného možného toku v síti nemůže být větší, než je kapacita řezu $W(M)$, a tedy ani větší než velikost toku x. Tok x je tedy maximální. Podobně řez $W(M)$ je minimální, protože žádný jiný řez nemůže mít větší kapacitu, než je velikost toku x.

Věta 2.33. (Ford-Fulkersonova věta o maximálním toku a minimálním řezu)
Velikost maximálního toku z počátečního do cílového uzlu je rovna kapacitě minimálního řezu oddělujícího tyto dva uzly.

Poznámka:
Větu ponecháme bez důkazu, protože je přímý a lze ho nalézt v [9].

3. Úvod do matematického programování

Matematické programování je oblast matematiky zabývající se hledáním optimálních řešení. Cílem je zpravidla minimalizovat nebo maximalizovat tzv. účelovou funkci, která vyjadřuje záměr nebo cíl, kterého bychom chtěli dosáhnout. Při hledání optimálního řešení jsme limitováni omezeními, která zaznamenáváme ve tvaru rovnic a nerovnic. Základy matematického programování položil G. B. Dantzig v polovině 20. století, viz [19]. Od té doby prošlo matematické programování rozsáhlou evolucí a používá se v mnoha oblastech od strojírenství, přes energetiku až po logistiku.

Matematické programování se vzhledem k aplikacím, složitostem účelových funkcí, omezení a charakteru proměnných dělí na několik kategorií, které zpravidla rozlišujeme z důvodu, že každá vyžaduje trochu jiné metody řešení. Pro logistické aplikace, kterým se ve své práci věnuji, jsou nejdůležitější následující dvě kategorie. První z nich jsou *deterministické modely*, u nichž jsou všechny hodnoty a vztahy pevně dané. Druhou kategorii představují *stochastické modely* reprezentující takové modely, kde je alespoň jedna veličina náhodnou proměnnou. Další klasifikaci modelů uvádí například [19]. Optimalizační úlohu lze obecně zapsat podle následující definice, viz [30].

Definice 3.1. (Úloha matematického programování)
Nechť $S \subset \mathbb{R}^n$ a $f : \mathbb{R}^n \to \mathbb{R}$, $n \in \mathbb{N}$. Pak úlohu matematického programování definujeme jako:

$$\min_{\mathbf{x}} \{f(\mathbf{x}) | \mathbf{x} \in S\},$$

kde S je množina přípustných řešení splňujících daná omezení a f je účelová funkce.

Ještě než se v další části textu zaměříme na úlohy lineárního programování, uvedeme některé pojmy z [2] a [30], které budou souviset s řešením základních modelů v kapitole 5.

Definice 3.2. (Lokální minimum)
Nechť $f : S \to \mathbb{R}$ je funkce s definičním oborem $S \subset \mathbb{R}^n$. Pak $\mathbf{x}_{\min} \in S$ je lokální neostré minimum právě tehdy, když existuje okolí $O(\mathbf{x}_{\min})$ bodu $\mathbf{x}_{\min}$ takové, že pro všechny body $\mathbf{x} \in S \cap O(\mathbf{x}_{\min})$ platí $f(\mathbf{x}_{\min}) \leq f(\mathbf{x})$. Pokud nahradíme $\leq$ symbolem $<$ a $O(\mathbf{x}_{\min})$ nahradíme $O(\mathbf{x}_{\min}) \setminus \{\mathbf{x}_{\min}\}$ mluvíme o ostrém lokálním minimu.

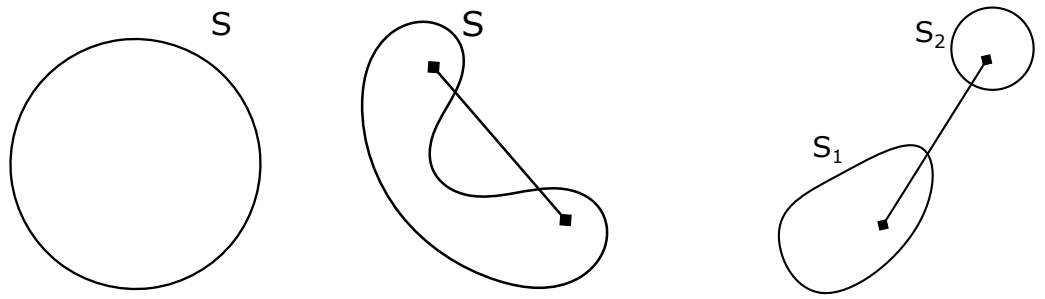
Definice 3.3. (Globální minimum)
Nechť $f : S \to \mathbb{R}$ je funkce s definičním oborem $S \subset \mathbb{R}^n$. Pak $\mathbf{x}_{\min} \in S$ je globální neostré minimum právě tehdy, když $\forall \mathbf{x} \in S$ platí $f(\mathbf{x}_{\min}) \leq f(\mathbf{x})$. Pokud nahradíme $\leq$ symbolem $<$, mluvíme o ostrém globálním minimu.

Obdobné definice lze uvést pro neostrá i ostrá, lokální i globální maxima v případě, že obrátíme nerovnosti, viz [30].

Věta 3.4. (Weierstrassova věta)
Mějme množinu $S \subset \mathbb{R}^n$, která je neprázdná, uzavřená a omezená a dále mějme spojitou funkci $f : S \to \mathbb{R}$. Pak úloha matematického programování $\min_{\mathbf{x}} \{f(\mathbf{x}) | \mathbf{x} \in S\}$ nabývá svého globálního minima, které budeme značit $\mathbf{x}_{\min}$.

Definice 3.5. (Konvexní množina)
Mějme množinu $S \subset \mathbb{R}^n$. Řekneme, že S je konvexní právě tehdy, když pro libovolné dva body $\mathbf{x}_1, \mathbf{x}_2 \in S$ a $\forall \lambda \in <0,1>$ platí $\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2 \in S$.

Poznámka:
Konvexní množina představuje důležitý matematický pojem, protože z její definice vyplývá, že s každými dvěma body $\mathbf{x}_1$ a $\mathbf{x}_2$ dané množiny S musí v příslušné množině ležet i všechny body úsečky, která body $\mathbf{x}_1$ a $\mathbf{x}_2$ spojuje, viz obr. 3.1. Všechny body úsečky spojující body $\mathbf{x}_1$ a $\mathbf{x}_2$ lze získat pomocí konvexní kombinace.



(a) Konvexní množina S (b) Nekonvexní množina S (c) Nekonvexní množina $S = S_1 \cup S_2$

Obrázek 3.1: Příklady množin ilustrujících konvexnost.

Definice 3.6. (Konvexní kombinace)
Mějme body $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_k \in \mathbb{R}^n$. Pak bod $\mathbf{x}$ splňující $\mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j$, kde $\sum_{j=1}^{k} \lambda_j = 1$ a $\forall j \in \{1, ..., k\} : \lambda_j \geq 0$ se nazývá konvexní kombinace bodů $\mathbf{x}_1, ..., \mathbf{x}_k$.

Definice 3.7. (Konvexní obal)
Nechť $S \subset \mathbb{R}^n$, pak řekneme, že $\mathbf{x} \in \mathbb{R}^n$ náleží do konvexního obalu množiny S, tj. $\mathbf{x} \in \text{conv}S \Leftrightarrow \exists\, k \in \mathbb{N}, \exists\, \lambda_1, ..., \lambda_k \geq 0, \sum_{j=1}^{k} \lambda_j = 1, \exists\, \mathbf{x}_1, ..., \mathbf{x}_k \in S : \mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j$.

Definice 3.8. (Konvexní funkce)
Nechť $S \subset \mathbb{R}^n, S \neq \emptyset$ je konvexní množina, $f : S \to \mathbb{R}$. Funkci f nazveme konvexní na množině S právě tehdy, když $\forall \mathbf{x}_1, \mathbf{x}_2 \in S, \forall \lambda \in (0,1) : f(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1-\lambda)f(\mathbf{x}_2)$.
Funkci f nazveme ryze konvexní na množině S právě tehdy, když $\forall \mathbf{x}_1, \mathbf{x}_2 \in S, \mathbf{x}_1 \neq \mathbf{x}_2, \forall \lambda \in (0,1) : f(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) < \lambda f(\mathbf{x}_1) + (1-\lambda)f(\mathbf{x}_2)$.
Funkce f je konkávní (ryze konkávní) na S právě tehdy, když $-f$ je konvexní (ryze konvexní).

Definice 3.9. (Nadrovina, lineární poloprostor)
Nechť $\alpha \in \mathbb{R}, \mathbf{p} \in \mathbb{R}^n, \mathbf{p} \neq \mathbf{0}$. Nadrovina H v prostoru $\mathbb{R}^n$ je definována jako množina $H = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{p}^\top \mathbf{x} = \alpha\}$. Nadrovina zároveň definuje dva uzavřené lineární poloprostory $H^+ = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{p}^\top \mathbf{x} \geq \alpha\}$ a $H^- = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{p}^\top \mathbf{x} \leq \alpha\}$ nebo pokud nahradíme neostré rovnosti ostrými, definuje nadrovina dva otevřené lineární poloprostory.

Lemma 3.10. (Konvexnost nadrovin a poloprostorů)
Uvažujme $\alpha \in \mathbb{R}, \mathbf{p} \in \mathbb{R}^n$. Pak nadrovina $H = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{p}^\top \mathbf{x} = \alpha\}$ a poloprostor $H^+ = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{p}^\top \mathbf{x} \geq \alpha\}$ podle definice 3.9 uvedené výše jsou konvexní množiny.

Věta 3.11.
Nechť $S \subset \mathbb{R}^n$ je uzavřená a konvexní množina, pak S je průnikem poloprostorů obsahujících S.

Lemma 3.12. (Průnik konvexních množin)
Průnik libovolného množství konvexních množin je konvexní množina.

Věta 3.13. (Konvexnost a globální minimum)
Nechť $S \subset \mathbb{R}^n, S \neq \emptyset$ je konvexní množina a $f : S \to \mathbb{R}$ je konvexní funkce na S. Pokud $\mathbf{x}_{\min}$ je lokálním minimem na množině $\{f(\mathbf{x})|\mathbf{x} \in S\}$ pak je také globálním minimem. Pokud je $\mathbf{x}_{\min}$ ostrým lokálním minimem, je také ostrým globálním minimem.

Poznámka:
Uvedli jsme pojmy, které budeme dále využívat při práci s optimalizačními úlohami. V navazující části 3.1 uvedeme úlohy lineárního programování. V následující podkapitole 3.2 se zaměříme na úlohu o maximálním toku v síti. Spolu s ní uvedeme i některé další související úlohy o tocích v síti. Navážeme odvozením duální úlohy k uvedené úloze o maximálním toku. V závěru kapitoly se budeme věnovat dvoustupňovým úlohám v 3.3 a deterministickým reformulacím stochastických úloh v 4.1.

3.1. Úlohy lineárního programování

Lineární programování pro nás bude výchozím bodem na cestě za složitějšími modely. Úlohu lineárního programování obecně zapíšeme jako

$$\begin{array}{lccccccccc}
\min & c_1x_1 & + & c_2x_2 & + & \cdots & + & c_nx_n & & \\
\text{za podmínek} & a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & \star & b_1, \\
& a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & \star & b_2, \\
& & & & \vdots & & & & & \\
& a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mn}x_n & \star & b_m, \\
& & & & & & x_1, x_2, & \ldots, x_n & \geq & 0,
\end{array}$$

kde $\star \in \{=, \leq, \geq\}$. Nadále budeme z důvodu přehlednosti výkladu uvažovat pouze jeden z těchto symbolů, a to $\leq$. Samozřejmě úlohu lze modifikovat i tak, že lze použít kterýkoli z těchto symbolů, ale o tom se zmíníme až později.

Funkce $z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$ je účelová funkce, kterou minimalizujeme. Zadané koeficienty $c_1, \ldots, c_n$ často reprezentují jednotkové ceny vztažené k rozhodovacím proměnným $x_1, \ldots, x_n$, jejichž hodnoty se snažíme určit. Každou podmínku vyjadřujeme ve formě nerovnic $\sum_{j=1}^{n} a_{ij}x_{ij} \leq b_i$. Koeficienty b_i dávají dohromady vektor $\mathbf{b}$, tedy takzvaný vektor pravé strany vyjadřující maximální zdroje, které lze při minimalizaci využít. Koeficienty a_{ij} tvoří matici známých koeficientů $\mathbf{A}$.

Proměnné $x_1, \ldots, x_n$ uvažujeme nezáporné. Pokud všechna x_i splňují příslušná omezení, v nichž se vyskytují, pak vektor $\mathbf{x}$ poskytuje přípustné řešení. Množinu takových řešení pak nazýváme množinou přípustných řešení.

Zápis výše lze uvažovat i ve stručnější formě takzvaného maticového zápisu následujícím způsobem.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \ \mathbf{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}, \ \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}, \ \mathbf{0} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}, \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix},$$

kde $\mathbf{0}$ je vektor délky n vyjadřující dolní mez pro vektor proměnných $\mathbf{x}$. Úlohu lineárního programování pak lze kompaktně zapsat v maticovém tvaru jako

$$\begin{array}{lll} \min & \mathbf{c}^\top \mathbf{x}, & \\ \text{za podmínek} & \mathbf{A}\mathbf{x} & \geq \mathbf{b}, \\ & \mathbf{x} & \geq \mathbf{0}. \end{array}$$

Jak jsme předeslali dříve, nejprve si ukážeme, jak nerovnosti v omezeních převést na rovnosti. Uvažujme libovolné i-té omezení, které lze zapsat $\sum_{j=1}^{n} a_{ij}x_j \circ b_i$, kde symbol $\circ \in \{\leq, \geq\}$. Takovou nerovnici můžeme snadno převést na rovnici zavedením nezáporné proměnné, kterou budeme nazývat *doplňková proměnná*, následujícími způsoby:

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij}x_j - x_{n+1} = b_i,$$
$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad \Leftrightarrow \quad \sum_{j=1}^{n} a_{ij}x_j + x_{n+1} = b_i.$$

Až doposud jsme uvažovali, že proměnné $x_1, ..., x_n$ jsou nezáporné. To nemusí být nutně pravda. Představme si triviální omezení $x \leq h$, kde h představuje horní mez, která je záporná. Obdobně může nastat situace, kdy $x \geq d$, kde d je dolní mez, která je taktéž záporná. Ve druhém případě může sice nastat situace, kdy x nabývá kladných hodnot, ale v prvním nikoli. S takovou situací se vypořádáme zavedením nové proměnné $\bar{x}$ takto

$$x \geq d \quad \Leftrightarrow \quad \bar{x} = x - d \geq 0,$$
$$x \leq h \quad \Leftrightarrow \quad \tilde{x} = h - x \geq 0.$$

Zbývá zmínit případ, kdy proměnná x není ohraničená, tj. $x \in \mathbb{R}$. V takovém případě stačí uvažovat součet dvou nezáporných proměnných, tj. $x = x^+ - x^-$, kde $x^+, x^- \geq 0$.

Poslední možnou úpravou úlohy lineárního programování je převedení minimalizační úlohy na maximalizační, kterou lze provést následujícím způsobem

$$\min \mathbf{c}^\top \mathbf{x} = -\max(-\mathbf{c}^\top \mathbf{x}).$$

Z výše uvedených transformací lze usuzovat, že úlohu lineárního programování lze zapsat mnoha způsoby.

Pro řešení je vhodný zápis úlohy ve *standardním tvaru*. Úloha v takovém tvaru má všechny proměnné nezáporné a navíc jsou všechna omezení zapsána ve tvaru rovnic. Úlohu ve standardním tvaru pak lze řešit simplexovou metodou. Uvedená metoda je implementována v systému GAMS, který používáme při řešení úloh v kapitole 5, metodu proto více nepopisujeme. Více o této metodě například v [5], [19].

Pokud jsou všechna omezení ve tvaru $\geq$, všechny proměnné jsou nezáporné a naše úloha lineárního programování je minimalizační, pak lze říci, že jde o úlohu v *kanonickém tvaru*. Naopak maximalizační úloha je v kanonickém tvaru, jsou-li všechna omezení ve tvaru $\leq$ a všechny proměnné jsou nezáporné. Všechny zmíněné zápisy úloh jsou shrnuty v následující tabulce převzaté z [2].

	Minimalizace	Maximalizace
Standardní tvar	$\min \mathbf{c}^\top\mathbf{x}$ za podmínek $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$	$\max \mathbf{c}^\top\mathbf{x}$ za podmínek $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$
Kanonický tvar	$\min \mathbf{c}^\top\mathbf{x}$ za podmínek $\mathbf{Ax} \geq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$	$\max \mathbf{c}^\top\mathbf{x}$ za podmínek $\mathbf{Ax} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$

Ještě než přikročíme k dualitě, uvedeme důležité vlastnosti úloh lineárního programování, které čerpáme z [30].

Věta 3.14. (Konvexnost úloh lineárního programování)
Nechť $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{c} \in \mathbb{R}^n$ a $\mathbf{b} \in \mathbb{R}^m$ pak množina přípustných řešení $S = \{\mathbf{x} \in \mathbb{R}^n | \mathbf{Ax} \leq \mathbf{b}\}$ a množina optimálních řešení úlohy lineárního programování $\min\limits_{\mathbf{x}}\{\mathbf{c}^\top\mathbf{x}|\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ jsou konvexní množiny.

Definice 3.15. (Polytop)
Nechť $k \in \mathbb{N}$ a $\mathbf{x}_1, ..., \mathbf{x}_k \in \mathbb{R}^n$. Pak $\text{conv}\{\mathbf{x}_1, ..., \mathbf{x}_k\}$ se nazývá polytop.

Definice 3.16. (Polyedrická množina)
Množina $S \subset \mathbb{R}^n$ se nazývá polyedrická, jestliže je průnikem konečného počtu uzavřených poloprostorů, tj. $\exists\, k \in \mathbb{N}, \exists\, \alpha_i, \exists\, \mathbf{p}_i \in \mathbb{R}^n, \mathbf{p}_i \neq \mathbf{0}, i = 1, ..., k : S = \cap_{i=1}^k \{\mathbf{x}|\mathbf{p}_i^\top\mathbf{x} \leq \alpha_i\}$.

Poznámka:
Z výše uvedených definic plynou následující vlastnosti. Každý polytop je polyedrická množina. Omezená polyedrická množina je polytop. Množina přípustných řešení úlohy lineárního programování je polyedrická množina.

Definice 3.17. (Krajní bod)
Mějme neprázdnou konvexní množinu $S \subset \mathbb{R}^n$. Bod $\mathbf{x} \in S$ nazveme krajním bodem množiny S právě tehdy, když $\forall\, \mathbf{x}_1, \mathbf{x}_2 \in S$ a $\lambda \in (0, 1)$ platí

$$\mathbf{x} = \lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \Rightarrow \mathbf{x} = \mathbf{x}_1 = \mathbf{x}_2.$$

Věta 3.18. (Existence krajního bodu)
Nechť $S = \{\mathbf{x}|\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\} \neq \emptyset$, kde $\mathbf{A} \in \mathbb{R}^{m\times n}$ je matice typu $m \times n$ s plnou řádkovou hodností, tj. $h(\mathbf{A}) = m$, pak množina S má alespoň jeden krajní bod.

Definice 3.19. (Směr)
Nechť $S \subset \mathbb{R}^n$ je neprázdná, uzavřená a konvexní množina. Pak $\mathbf{d} \in \mathbb{R}^n, \mathbf{d} \neq \mathbf{0}$ se nazývá směr množiny S, jestliže platí $\mathbf{x} \in S, \forall\lambda \geq 0 : \mathbf{x} + \lambda\mathbf{d} \in S$.
Dále řekneme, že $\mathbf{d}_1, \mathbf{d}_2$ jsou různé směry množiny S, jestliže $\forall\alpha > 0$ platí $\mathbf{d}_1 \neq \alpha\mathbf{d}_2$.

Definice 3.20. (Krajní směr)
Směr $\mathbf{d}$ množiny S se nazývá krajní směr množiny S, jestliže $\forall\mathbf{d}_1, \mathbf{d}_2$ směry množiny S a $\forall\lambda_1, \lambda_2 > 0$ platí

$$\mathbf{d} = \lambda_1\mathbf{d}_1 + \lambda_2\mathbf{d}_2 \Rightarrow \exists\alpha > 0 : \mathbf{d}_1 = \alpha\mathbf{d}_2.$$

Věta 3.21. (Existence krajního směru)
Nechť $S = \{\mathbf{x} \in \mathbb{R}^n|\mathbf{Ax} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ a $h(\mathbf{A}) = m$, pak S má alespoň jeden krajní směr. Množina S není ohraničená.

Poznámka:
Následující věta říká, že v úloze lineárního programování ve standardním tvaru lze každý bod množiny přípustných řešení vyjádřit jako součet konvexní kombinace krajních bodů množiny S a nezáporné lineární kombinace krajních směrů.

Věta 3.22. (Věta o reprezentaci)
Nechť $S \subset \mathbb{R}^n, S \neq \emptyset, S = \{\mathbf{x}|\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, $h(\mathbf{A}) = m$. Krajní body množiny S označíme $\mathbf{x}_1, ..., \mathbf{x}_k$ a krajní směry označíme $\mathbf{d}_1, ..., \mathbf{d}_l$. Pak $\mathbf{x} \in S \Leftrightarrow \exists \mu_j \geq 0, j = 1, ..., l, \exists \lambda_j \geq 0, j = 1, ..., k : \sum_{j=1}^{k} \lambda = 1 \wedge \mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j + \sum_{j=1}^{l} \mu_j \mathbf{d}_j$.

Věta 3.23. (Podmínky optimality)
Mějme úlohu lineárního programování ve standardním tvaru. Označíme $S = \{\mathbf{x}|\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ množinu přípustných řešení a množinu D všech optimálních řešení úlohy $\min_{\mathbf{x}}\{\mathbf{c}^\top \mathbf{x}|\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. Předpokládáme matici $\mathbf{A}$ s plnou řádkovou hodností a neprázdnou množinu S. Uvažujeme krajní body $\mathbf{x}_1, ..., \mathbf{x}_k$ a krajní směry $\mathbf{d}_1, ..., \mathbf{d}_l$. Pak $D \neq \emptyset$ právě tehdy, když $\forall j \in \{1, ..., l\} : \mathbf{c}^\top \mathbf{d}_j \geq 0$. Navíc pokud $D \neq \emptyset$, pak $\exists\ \mathbf{x}_i \in D$, který je krajním bodem množiny S.

Důkaz:
Využijeme větu 3.22. Nechť $S \subset \mathbb{R}^n$ je množina přípustných řešení. Všechna $\mathbf{x} \in \mathbb{R}^n$: $\mathbf{x} \in S \Leftrightarrow \exists\ \lambda_j \geq 0, j = 1, ..., k, \sum_{j=1}^{k} \lambda_j = 1$ a $\exists\ \mu_j \geq 0, j = 1, ..., l$ tak, že $\mathbf{x} = \sum_{j=1}^{k} \lambda_j \mathbf{x}_j + \sum_{j=1}^{l} \mu_j \mathbf{d}_j$. Díky tomu lze úlohu lineárního programování zapsat s užitím této reprezentace $\mathbf{x} \in S$ následujícím způsobem:

$$\min_{\lambda,\mu}\Big\{\mathbf{c}^\top\Big(\sum_{j=1}^{k}\lambda_j\mathbf{x}_j + \sum_{j=1}^{l}\mu_j\mathbf{d}_j\Big)\Big|\lambda_j \geq 0, j = 1, ..., k, \sum_{j=1}^{k}\lambda_j = 1, \mu_j \geq 0, j = 1, ..., l\Big\}.$$

Pokud $\mathbf{c}^\top \mathbf{d}_j < 0$ pro nějaké j, pak $\mu_j \to \infty$ (zbývající μ_j jsou rovna 0) a účelová funkce je neohraničená zdola. Tedy nutná a postačující podmínka je jasná. Pokud $\forall j : \mathbf{c}^\top \mathbf{d}_j \geq 0$, pak jelikož minimalizujeme, přiřadíme $\forall j : \mu_j = 0$. Pak řešíme úlohu

$$\min_{\lambda,\mu}\Big\{\mathbf{c}^\top \sum_{j=1}^{k}\lambda_j\mathbf{x}_j\Big|\lambda_j \geq 0, j = 1, ..., k, \sum_{j=1}^{k}\lambda_j = 1\Big\}.$$

To lze udělat nastavením $\lambda_i = 1$ pro i tak, že $\min\{\mathbf{c}^\top \mathbf{x}_j|j = 1, ..., k\} = \mathbf{c}^\top \mathbf{x}_i$ (ostatní λ_j jsou nulová).

Poznámka:
Z věty 3.23 plyne hlavní myšlenka řešení úlohy lineárního programování výše zmíněnou simplexovou metodou, kterou je postupování od krajního bodu ke krajnímu bodu a zlepšování hodnoty účelové funkce.

Dualita

Každá úloha lineárního programování je spjata s jinou úlohou lineárního programování, která je původní úlohou jednoznačně určena, viz např. [19]. Takto sdružené úlohy nazýváme *dvojice duálně sdružených úloh*, navíc platí, že duální úloha k duální úloze je úloha

primární. Původní úlohu budeme nazývat primární, k ní sdruženou úlohu pak duální. Pojem *duální* pochází z lineární algebry, viz [2], zatímco pojem *primární* byl navržen Tobiasem Dantzigem, otcem G. B. Dantziga, který byl rovněž matematikem, jako náhrada za sousloví popisující primární úlohu - „původní úloha, k níž je tato duální“, viz [2]. Vlastnosti duálních úloh budeme v dalším textu využívat, a proto je dále postupně uvádíme. Vztah primární a duální úlohy popisuje následující tabulka, viz [2].

	Primární úloha		Duální úloha	
Standardní tvar	min za podmínek	$\mathbf{c}^\top\mathbf{x}$ $\mathbf{Ax} = \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	max za podmínek	$\mathbf{b}^\top\mathbf{w}$ $\mathbf{A}^\top\mathbf{w} \leq \mathbf{c}$ $\mathbf{w}$ neomezené
Kanonický tvar	min za podmínek	$\mathbf{c}^\top\mathbf{x}$ $\mathbf{Ax} \geq \mathbf{b}$ $\mathbf{x} \geq \mathbf{0}$	max za podmínek	$\mathbf{b}^\top\mathbf{w}$ $\mathbf{A}^\top\mathbf{w} \leq \mathbf{c}$ $\mathbf{w} \geq \mathbf{0}$

Tabulka 3.1: Zápis primárních a duálních úloh ve standardním a kanonickém tvaru.

Pokud máme obecnou úlohu lineárního programování, která není ani v kanonickém ani ve standardním tvaru, převedeme ji na standardní nebo kanonický tvar a pak určíme příslušnou duální úlohu. Principy, které se uplatňují při tvorbě duální úlohy jsou shrnuty v následující tabulce, viz [19].

Maximalizační úloha	Minimalizační úloha
primární	duální
duální	primární
omezení typu $\leq$	nezáporná proměnná
omezení typu $\geq$	nekladná proměnná
omezení typu rovnice	neomezená proměnná
nezáporná proměnná	omezení typu $\geq$
nekladná proměnná	omezení typu $\leq$
neomezená proměnná	omezení typu rovnice

Tabulka 3.2: Vlastnosti primárních a duálních úloh.

Primární a duální úloha spolu souvisí nejen zápisem, ale i svými vlastnostmi. Vztahy primární a duální úlohy lze pak popsat s pomocí následujících vět, které čerpáme z [19].

Věta 3.24. (Slabá věta o dualitě)
Nechť primární úloha je minimalizační a duální úloha je maximalizační, viz tabulka 3.1. Nechť $\mathbf{x}$ je libovolné přípustné řešení primární úlohy a $\mathbf{w}$ je libovolné přípustné řešení duální úlohy. Pak platí

$$\mathbf{c}^\top\mathbf{x} \geq \mathbf{b}^\top\mathbf{w}.$$

Hodnota účelové funkce minimalizační úlohy reprezentuje ve všech bodech své množiny přípustných řešení horní mez pro všechny hodnoty účelové funkce maximalizační úlohy. Opačně také platí, že hodnoty maximalizační účelové funkce ve všech bodech množiny přípustných řešení tvoří dolní mez pro hodnoty účelové funkce minimalizační duální úlohy.

Důsledek 3.25.
Je-li množina přípustných řešení primární minimalizační úlohy neprázdná a je-li účelová

funkce této úlohy zdola neomezená, pak duálně sdružená úloha nemá žádné přípustné řešení.

Důsledek 3.26.
Je-li množina přípustných řešení duální maximalizační úlohy neprázdná a je-li účelová funkce této úlohy shora neomezená, pak duálně sdružená primární úloha nemá žádné přípustné řešení.

Důsledek 3.27.
Nemá-li jedna z dvojice duálně sdružených úloh přípustné řešení, pak druhá úloha nemá optimální řešení.

Věta 3.28. (Věta o silné dualitě)
Má-li jedna z duálně sdružených úloh optimální řešení, má optimální řešení i duální úloha a navíc, optimální hodnoty účelových funkcí jsou si rovny.

Věta 3.29. (Věta o komplementaritě)
Přípustná řešení primární a duální úlohy

$$\begin{aligned} \min \mathbf{c}^\top \mathbf{x}, \quad & \max \mathbf{b}^\top \mathbf{w}, \\ \mathbf{A}\mathbf{x} \geq \mathbf{b}, \quad & \mathbf{A}^\top \mathbf{w} \leq \mathbf{c}, \\ \mathbf{x} \geq \mathbf{0}, \quad & \mathbf{w} \geq \mathbf{0}, \end{aligned}$$

jsou optimální právě tehdy, když platí

$$x_j(\sum_{i=1}^{m} a_{ij}w_i - c_j) = 0, \qquad \forall j = 1, ..., n,$$
$$w_i(\sum_{j=1}^{n} a_{ij}x_j - b_i) = 0, \qquad \forall i = 1, ..., m.$$

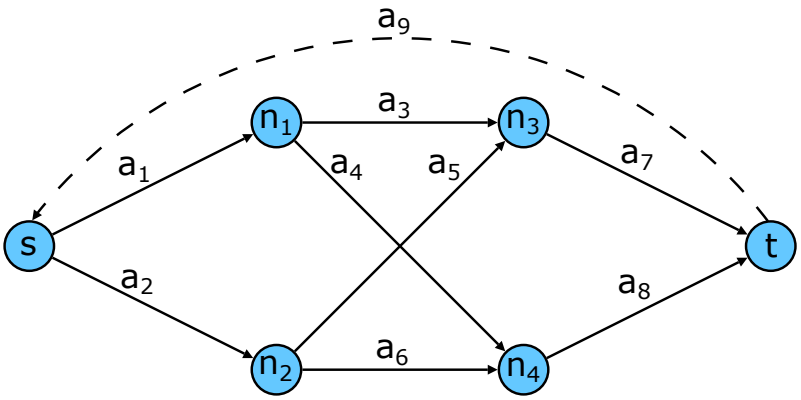
Rovnosti uvedené výše platí pro duálně sdružené úlohy v kanonickém tvaru, viz tabulka 3.1. To znamená, že jestliže má nějaká proměnná kladnou hodnotu, odpovídající duálně sdružené omezení musí být splněno jako rovnost. Naopak, jestliže je nějaké omezení splněno jako ostrá nerovnost, pak odpovídající duálně sdružená proměnná musí být nulová.

3.2. Úloha o maximálním toku a duální úloha

Mezi typické úlohy lineárního programování patří také úlohy o toku v síti. Předpokládáme, že v síti se nevyskytují smyčky. Lze sice uvažovat i sítě se smyčkami, ale ty u úloh o tocích v síti nejsou při řešení využívány. Rád bych zdůraznil, že zde budeme s ohledem na následující modely v kapitole 5 využívat modifikované značení převzaté především z literatury zabývající se toky v sítích z pohledu lineárního programování, viz [2], [7], [13], a modely pro napadání sítí, viz [18], [33], [37].

Dále uvažujeme úlohu o maximálním toku, viz předchozí kapitola 2. Nyní ji však budeme formulovat jako úlohu lineárního programování. Mějme síť S dle definice 2.21 se dvěma významnými různými uzly, počátečním uzlem s a cílovým uzlem t tak, jak jsme je definovali v definici sítě. Předpokládejme zároveň, že neexistuje přímá hrana z počátečního

uzlu s do cílového uzlu t, protože by tato hrana implikovala triviální část řešení spočívající v tom, že bychom mohli přímo určit část maximálního toku jako úplné využití této hrany. Cílem je maximalizovat tok z počátečního do cílového uzlu, viz definice 2.31. Jediný limit představuje propustnost sítě, respektive kapacity námi definovaných hran. Než přikročíme k formálně zapsanému modelu, vše ukážeme na příkladu. Uvažujme následující síť na obrázku 3.2:



Obrázek 3.2: Testovací síť.

Uvedená síť se skládá z množiny uzlů $N = \{s, n_1, n_2, n_3, n_4, t\}$, množiny hran $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\}$ a pomocné návratové hrany a_9, která nemusí být součástí zadání původní úlohy. Pokud v zadání, obdobně jako v našem případě, není, lze ji bez újmy na obecnosti v případě potřeby do sítě doplnit, viz obrázek 3.2. Důvod přidání návratové hrany se objasní při řešení příkladu. K uvedenému grafu sestavíme s využitím matice incidence dle definice 2.19 a s využitím vektoru tokových proměnných $\mathbf{x}$ dle definice 2.22 úlohu o maximálním toku v síti takto:

$$\max \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{s1} & x_{s2} & x_{13} & x_{14} & x_{23} & x_{24} & x_{3t} & x_{4t} & x_{ts} \end{pmatrix}^\top,$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_{s1} \\ x_{s2} \\ x_{13} \\ x_{14} \\ x_{23} \\ x_{24} \\ x_{3t} \\ x_{4t} \\ x_{ts} \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ u_{s1} \\ u_{s2} \\ u_{13} \\ u_{14} \\ u_{23} \\ u_{24} \\ u_{3t} \\ u_{4t} \\ u_{ts} \end{pmatrix},$$

$$\mathbf{x} \geq \mathbf{0}.$$

Aby bylo jednoznačné, o kterých konkrétních hranách v případě toku mluvíme, zavádíme pro tok na hranách vektor proměnných $\mathbf{x} = \{x_{s1}, x_{s2}, x_{13}, x_{14}, x_{23}, x_{24}, x_{3t}, x_{4t}, x_{ts}\}$.

Každá proměnná se vztahuje k jedné konkrétní hraně, tedy například k hraně a_1 se vztahuje proměnná x_{s1}. Pořadí indexů u proměnných x_{ij} značí orientaci hrany, tedy v případě x_{s1} hrana vychází z počátečního uzlu s a končí v uzlu n_1. Pro zjednodušení budeme indexy uzlů používat také jako indexy příslušných proměnných. Kapacity hran, které budou přiřazovány k hranám stejně jako proměnné x_{ij}, označíme u_{ij}. Protože uvažujeme cirkulaci, budeme úlohu maximálního toku řešit jako maximalizaci toku přes návratovou hranu. Kirchhoffovy zákony z definice 2.22 tedy platí pro všechny uzly.

Ukázali jsme, že úlohu o maximálním toku v síti lze zapsat jako úlohu lineárního programování. Připomeňme, že proměnné x_{ij} značí toky na hranách a u_{ij} jsou kapacity hran. Uvažujme podmnožinu množiny uzlů $N' = N \setminus \{s, t\}$. Matematický model můžeme formálně zapsat:

$$\max x_{ts},$$

za podmínek

$$\sum_{j \in N'} x_{sj} - \sum_{i \in N'} x_{is} - x_{ts} \leq 0, \tag{3.1}$$

$$\sum_{j \in N} x_{nj} - \sum_{i \in N} x_{in} \leq 0, \qquad \forall n \in N', \tag{3.2}$$

$$\sum_{j \in N'} x_{tj} - \sum_{i \in N'} x_{it} + x_{ts} \leq 0, \tag{3.3}$$

$$0 \leq x_{ij} \leq u_{ij}, \qquad \forall (i,j) \in A. \tag{3.4}$$

Úloha o maximálním toku má tvar, který je úpravou tvaru z [37]. Jestliže je naším záměrem maximalizace toku v síti, stačí maximalizovat tok přes pomocnou návratovou hranu x_{ts}, jejíž kapacitu u_{ts} nastavíme na dostatečně vysokou hodnotu tak, aby nelimitovala možný tok v síti, například na součet všech kapacit hran v síti. Přestože formálně maximalizujeme tok přes jednu hranu, účelová funkce má tvar $\mathbf{c}^\top \mathbf{x}$, kde vektor $\mathbf{c}^\top = (0 \dots 0\, 1)$. Místo toku v síti uvažujeme de facto cirkulaci v síti, která na výsledku problému nic nemění, ale zjednoduší účelovou funkci na pouhou jednu proměnnou. Další variantou, kterou by bylo možné použít, je maximalizace součtu toků v hranách začínajících v počátečním uzlu s, nebo končících v cílovém uzlu t. Omezení (3.1) - (3.3) by bylo možné s drobnou úpravou shrnout do jednoho omezení ve tvaru (3.2), pokud bychom indexy uvažovali z celé množiny uzlů N. Omezení jsou však rozepsána záměrně. Prvním důvodem je názornost a jasné uvedení pohledu na problém, druhým důvodem je využití takového zápisu při odvozování duální úlohy, kterou dále odvodíme s pomocí uvedeného příkladu, viz obrázek 3.2, a kterou budeme využívat v následující kapitole 5.

Omezením (3.1) - (3.3) odpovídá prvních šest řádků matice v příkladu k obrázku 3.2. Zbylá část matice tvoří jednotkovou submatici odpovídající kapacitním omezením (3.4). Všechna omezení jsou tedy zapsána v maticovém tvaru

$$\begin{pmatrix} \mathbf{A} \\ \mathbf{I} \end{pmatrix} \mathbf{x} \leq \begin{pmatrix} \mathbf{b} \\ \mathbf{u} \end{pmatrix}, \mathbf{x} \geq \mathbf{0},$$

kde $\mathbf{A}$ je incidenční matice, viz definice 2.19. Při hlubším zamyšlení nad omezeními (3.1) - (3.3) z jejich zápisu vyplývá, že je teoreticky možné, aby byl přítok do uzlu větší než odtok. Pokud si ale představíme analogii se systémem potrubí, zjistíme, že taková

úvaha ničemu nevadí. Zužující se potrubí zkrátka nepropustí větší množství toku, než povolí kapacita nejužšího potrubí. Navíc uvažujeme cirkulaci. Přirozeně tedy musí být tok v síti konstantní a nemůže se s časem měnit.

Nyní odvodíme duální úlohu podle tabulky 3.1 z předchozí kapitoly 3. Díky tomu, jakým způsobem jsme maticově zapsali všechna omezení, se můžeme stále držet příslušné tabulky. Matici $\binom{\mathbf{A}}{\mathbf{I}}$ transponujeme, vyměníme vektor pravé strany $\binom{\mathbf{b}}{\mathbf{u}}$ s vektorem $\mathbf{c}$ původní účelové funkce a místo původních proměnných x_{ij} zavedeme nezáporné duální proměnné α_i a θ_{ij}. Pro náš příklad dostaneme:

$$\min \left(\mathbf{0}^\top, \mathbf{u}^\top\right) \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\theta} \end{pmatrix},$$

$$\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\alpha_s \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_t \\ \theta_{s1} \\ \theta_{s2} \\ \theta_{13} \\ \theta_{14} \\ \theta_{23} \\ \theta_{24} \\ \theta_{3t} \\ \theta_{4t} \\ 0
\end{pmatrix}
\geq
\begin{pmatrix}
0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0
\end{pmatrix},$$

$$\boldsymbol{\alpha} \geq \mathbf{0}, \boldsymbol{\theta} \geq \mathbf{0},$$

kde $\boldsymbol{\alpha}^\top = (\alpha_s \dots \alpha_t)$, $\boldsymbol{\theta}^\top = (\theta_{s1} \dots \theta_{4t}\, 0)$, $\mathbf{u}^\top = \{u_{s1}\, u_{s2}\, u_{13}\, u_{14}\, u_{23}\, u_{24}\, u_{3t}\, u_{4t}\, u_{ts}\}$. Vektor proměnných α_i je duální k omezením (3.1) - (3.3) v primární úloze. Jelikož se omezení v primární úloze vztahovala vždy ke konkrétnímu uzlu, pak i α_i je proměnná svázaná s uzlem $n_i \in N$. Ze zápisu účelové funkce vidíme, že proměnné α_i neovlivňují její hodnotu. Proměnné θ_{ij} jsou duální proměnné ke kapacitním omezením z primární úlohy. Budeme se tedy snažit minimalizovat skalární součin vektoru proměnných $\boldsymbol{\theta}$ s vektorem kapacit $\mathbf{u}$. Proměnná $\theta_{ts} = 0$, viz věta o komplementaritě (věta 3.29), neboť primární hodnotu x_{ts} jsme v účelové funkci primární úlohy maximalizovali. V důsledku duality je pak ve vektoru pravé strany v poslední složce jednička. Stručněji zapsaná omezení našeho příkladu mají tvar:

$$\begin{aligned}
\alpha_s - \alpha_1 + \theta_{s1} &\geq 0, \\
\alpha_s - \alpha_2 + \theta_{s2} &\geq 0, \\
&\dots \\
\alpha_4 - \alpha_t + \theta_{4t} &\geq 0, \\
\alpha_t - \alpha_s &\geq 1.
\end{aligned}$$

Duální omezení k omezením z obecné primární úlohy lze pak kompaktně zapsat následujícím způsobem.

$$\begin{aligned} \alpha_i - \alpha_j + \theta_{ij} &\geq 0, \qquad \forall (i,j) \in A, \\ \alpha_t - \alpha_s &\geq 1. \end{aligned} \tag{3.5}$$

Nyní poskládáme vše dohromady. Maximalizace přejde v duální úloze na minimalizaci součtu skalárního součinu proměnných θ_{ij} s kapacitami hran u_{ij}. Následují omezení zapsaná výše. Proměnné α_i, θ_{ij} jsou nezáporné. Dostáváme tak obecně duální úlohu

$$\min \sum_{(i,j)\in A} u_{ij}\theta_{ij},$$

za podmínek

$$\begin{aligned} \alpha_i - \alpha_j + \theta_{ij} &\geq 0, \qquad \forall (i,j) \in A, && (3.6)\\ \alpha_t - \alpha_s &\geq 1, && (3.7)\\ \alpha_i &\geq 0, \qquad \forall i \in N, \\ \theta_{ij} &\geq 0, \qquad \forall (i,j) \in A. \end{aligned}$$

Zdůrazníme některé důležité vlastnosti této duální úlohy. Jelikož matice primární úlohy o maximálním toku je totálně unimodulární, viz [36], je matice duální úlohy rovněž totálně unimodulárni. Potom platí, viz [37], že pro celočíselné hodnoty pravých stran omezení je optimální řešení celočíselné. Navíc i všechny krajní body množiny přípustných řešení mají celočíselné souřadnice. Podle [37] jsou navíc tyto hodnoty binární. Řešení tedy můžeme hledat pomocí efektivní simplexové metody, ale úlohu zapíšeme z důvodu další interpretace pomocí $0-1$ proměnných, tedy

$$\min \sum_{(i,j)\in A} u_{ij}\theta_{ij},$$

za podmínek

$$\begin{aligned} \alpha_i - \alpha_j + \theta_{ij} &\geq 0, \qquad \forall (i,j) \in A, && (3.8)\\ \alpha_t - \alpha_s &\geq 1, && (3.9)\\ \alpha_i &\in \{0,1\}, \qquad \forall i \in N, \\ \theta_{ij} &\in \{0,1\}, \qquad \forall (i,j) \in A. \end{aligned}$$

Nyní tuto úlohu dále analyzujeme. Nezápornost proměnných α_i nám dovolí zvolit $\alpha_s = 0$. Navíc pro takto zdola omezené hodnoty proměnné α_i plyne z poslední nerovnosti, že lze uvažovat $\alpha_t = 1$.

Jak víme, minimální řez rozdělí graf na dvě komponenty N_s a N_t, kde N_s je komponenta obsahující počáteční uzel s a N_t je komponenta obsahující cílový uzel t, tj. $s \in N_s$, $t \in N_t$, $N_s \cap N_t = \emptyset$. Minimální řez je identifikován pomocí $\alpha_i = 1$, pokud se uzel i nachází v komponentě N_t, v opačném případě, pokud se nachází v komponentě N_s, pak $\alpha_i = 0$. Omezení (3.7) je možné nahradit rovnostmi $\alpha_t = 1$, $\alpha_s = 0$. Uvažujme nyní omezení (3.6), kde vystupuje proměnná θ_{ij} přes níž se snažíme minimalizovat účelovou funkci. Proměnná θ_{ij} má pak dvě dolní omezení ve tvaru $\theta_{ij} \geq 0$ a $\theta_{ij} \geq \alpha_j - \alpha_i$ (plyne z omezení

(3.6)), která vedou na $\theta_{ij} = \max\{\alpha_j - \alpha_i, 0\}$. S ohledem na hodnoty proměnné α_i lze říci, že proměnná $\theta_{ij} \in \{0, 1\}$, což nijak neporušuje omezení (3.6). Pokud $\theta_{ij} = 1$, pak hrana náleží do minimálního řezu, jinak je rovna nule. Toto tvrzení lze snadno ověřit uvažováním čtyř tříd hran - zaprvé $i \in N_s, j \in N_s$, zadruhé, $i \in N_s, j \in N_t$, zatřetí $i \in N_t, j \in N_s$ a začtvrté $i \in N_t, j \in N_t$.

Takové řešení námi formulované duální úlohy je přípustné a optimální, protože hodnota účelové funkce vyjadřující kapacitu minimálního řezu je díky platnosti věty o silné dualitě (viz věta 3.28) rovna hodnotě maximálního toku v síti.

3.3. Dvouúrovňové programování

V reálném životě nastává mnoho situací, kdy je potřeba uvažovat problém v několika úrovních rozhodování. Například pokud bychom měli umístit novou hasičskou stanici, bude naším cílem ji umístit tak, abychom minimalizovali maximální čas dojezdu k nejvzdálenějším obcím, tedy první úrovní našeho rozhodnutí by bylo vhodné umístění hasičské stanice, druhou úrovní pak odkud hasiči k požáru vyrazí. V případě rozmisťování mýtných bran na dálnicích chce stát maximalizovat své výdělky, zatímco motoristé chtějí minimalizovat své cestovní náklady. Navíc obecně při rozmisťování obranných prvků a kontrol se snaží bezpečnostní složky zpravidla minimalizovat maximální možné poškození, tedy v první úrovni se zjistí, kde jsou nejslabší místa a v druhé úrovni se k nim přiřadí vhodné obranné prvky.

My se v případě těchto úvah o dvouúrovňové optimalizaci zaměříme na konkrétní problém. Představme si uživatele sítě, jehož cílem je přepravit maximální možné množství materiálu skrz síť. Jediné jeho omezení představuje propustnost sítě, respektive jednotlivých hran. Na problém se lze také dívat tak, že kapacita hrany se odvíjí od kvality nějaké cesty, nebo ceny za přepravu přes ní. V takovém případě by například dálnice, kde se platí mýto, měla menší kapacitu než prostá silnice.

Na druhé straně si představme například policejní složky, kterým se nelíbí, že uživatel sítě převáží nedovolené zboží. Policejní složky se pochopitelně nemohou soustředit pouze na jeden problém, a proto jsou jejich kapacity značně omezené. Navíc policie velmi pravděpodobně vůbec netuší, kde se může výrobna zakázaného zboží (počáteční uzel s) nacházet stejně tak, jako netuší, kde se nachází koncový zákazník (cílový uzel t), nebo nemá dostatečné kapacity na hlídání všech uzlů a hran poblíž podezřelých míst. Jediným způsobem, jak uškodit uživateli, je pravidelná kontrola hran a uzlů v síti, viz [37].

Dostáváme se tak k jádru problému. Cílem policistů je minimalizovat maximální přepravené množství zakázaného zboží v síti. Takový model lze zapsat takto:

$$\min_{\boldsymbol{\gamma}} \max_{\mathbf{x}} x_{ts},$$

za podmínek

$$\sum_{j \in N'} x_{sj} - \sum_{i \in N'} x_{is} - x_{ts} \leq 0, \tag{3.10}$$

$$\sum_{j \in N} x_{nj} - \sum_{i \in N} x_{in} \leq 0, \qquad \forall n \in N', \tag{3.11}$$

$$\sum_{j \in N'} x_{tj} - \sum_{i \in N'} x_{it} + x_{ts} \leq 0, \tag{3.12}$$

$$x_{ij} - u_{ij}(1 - \gamma_{ij}) \leq 0, \qquad \forall (i,j) \in A, \tag{3.13}$$

$$\sum_{(i,j)\in A} r_{ij}\gamma_{ij} \leq R, \tag{3.14}$$

$$x_{ij} \geq 0, \qquad \forall (i,j) \in A \cup \{(t,s)\},$$

$$\gamma_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A, \tag{3.15}$$

kde $N' = N \setminus \{s,t\}$. Proměnné x_{ij} vyjadřují tok přepravovaného zboží na hraně (i,j), proměnné γ_{ij} představují zásah napadajícího do sítě, jsou-li rovny jedné. Konstanty r_{ij} zastávají roli cen napadení příslušných hran. V součtu pak platí, že cena za napadené hrany nesmí překročit rozpočet napadajícího R, viz omezení (3.14). Napadající se snaží minimalizovat maximální tok v síti a uživatel se snaží maximálně využít reziduální síť. Minimalizace představuje vnější problém, maximalizace pak vnitřní optimalizační úlohu, kterou bychom mohli řešit přímo, pokud bychom nějakým způsobem pevně zvolili hodnoty proměnné γ_{ij} a určili tak nové meze pro některé kapacity hran (i,j).

Napadající se snaží minimalizovat maximum lineární funkce místo toho, aby minimalizoval lineární funkci přímo. Podobné optimalizační funkce se objevují taktéž v teorii her, konkrétně ve hrách, které se nazývají Stackelbergovy hry, viz [24], [25].

Představme si tedy, že pevně zvolíme nějaké hodnoty γ_{ij}. V takovém případě přejde vnitřní problém na úlohu o maximálním toku a omezení (3.10) - (3.12) pak představují klasická omezení zachovávající tok v síti. Pro pevné hodnoty γ_{ij} vytvoří omezení (3.13) v případě některých hran (i,j) novou horní mez toku, jinak k zásadní změně úlohy nedochází.

Problémem zůstává teď už jen účelová funkce. Uvažujme následující model, kde místo vnitřní maximalizace toku v síti budeme uvažovat duální úlohu o minimálním řezu, viz podkapitola 3.2, a zahrneme binární proměnné α_i, θ_{ij}.

$$\min_{\boldsymbol{\gamma}} \min_{\boldsymbol{\alpha},\boldsymbol{\theta}} \sum_{(i,j)\in A} u_{ij}(1 - \gamma_{ij})\theta_{ij},$$

za podmínek

$$\alpha_i - \alpha_j + \theta_{ij} \geq 0, \qquad \forall (i,j) \in A,$$

$$\alpha_t - \alpha_s \geq 1,$$

$$\sum_{(i,j)\in A} r_{ij}\gamma_{ij} \leq R,$$

$$\alpha_i \in \{0,1\}, \qquad \forall i \in N,$$

$$\gamma_{ij}, \theta_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A.$$

Minimalizace maximální přepravené kapacity v síti skrz návratovou hranu přešla po aplikaci duality na minimalizaci minimálního řezu. Díky platnosti věty o silné dualitě platí, že čím menší je minimální řez, tím menší je maximální tok v síti. Charakter úlohy se tedy zachovává a s ním i optimální řešení, viz [37].

Účelová funkce je téměř stejná jako v případě úlohy o minimálním řezu, s výjimkou členu $(1-\gamma_{ij})$, který zastupuje zásah útočníka do sítě. Jestliže útočník přeruší hranu (i,j), bude hodnota $(1-\gamma_{ij})$ rovna nule, a tím pádem dochází ke zmenšení toku v síti. Omezení z vnitřní primární úlohy o maximálním toku jsme nahradili duálními, zbylý tvar úlohy

zůstává beze změny. Kapacitní omezení pro jednotlivé hrany se i zde projevují v účelové funkci opět ve formě členů θ_{ij}.

Účelová funkce není lineární, protože po roznásobení obsahuje součin neznámých proměnných $\theta_{ij}\gamma_{ij}$. Pro snazší řešení problému by bylo vhodné účelovou funkci nějakým způsobem linearizovat, aby bylo možné použít metody celočíselného lineárního programování. Toho lze docílit nahrazením $(1-\gamma_{ij})\theta_{ij}$ proměnnou β_{ij}, viz [37]. Tuto proměnnou lze uvažovat jako $\beta_{ij} \in \{0,1\}, \forall (i,j) \in A$ a navíc platí $\beta_{ij} \geq \theta_{ij} - \gamma_{ij}$. S užitím této substituce, lze zapsat předchozí model následujícím způsobem.

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\begin{aligned}
\alpha_i - \alpha_j + \theta_{ij} &\geq 0, && \forall (i,j) \in A, && (3.16)\\
\alpha_t - \alpha_s &\geq 1, \\
\beta_{ij} + \gamma_{ij} - \theta_{ij} &\geq 0, && \forall (i,j) \in A, && (3.17)\\
\sum_{(i,j)\in A} r_{ij}\gamma_{ij} &\leq R, \\
\alpha_i &\in \{0,1\}, && \forall i \in N, \\
\beta_{ij}, \gamma_{ij}, \theta_{ij} &\in \{0,1\}, && \forall (i,j) \in A.
\end{aligned}$$

Ještě než vysvětlíme význam jednotlivých omezení a případy, které mohou nastat při různých volbách hodnot proměnných, se zamyslíme nad tím, zda by nebylo možné z modelu některá omezení vynechat. Nejprve ukážeme, že omezení (3.17) lze nahradit rovností.

Uvažujme libovolné optimální řešení, kde $\theta_{ij} = 0$. Pro přípustnost řešení můžeme uvažovat libovolně β_{ij}, γ_{ij} rovny nule nebo jedné. Vzhledem k účelové funkci však víme, že $\beta_{ij} = 1$ zvýší hodnotu účelové funkce, a tak budeme volit $\beta_{ij} = 0$. Hodnotu γ_{ij} lze v tomto případě volit libovolně, protože jakoukoli volbou neporušíme omezení a zároveň nezměníme hodnotu účelové funkce. Obdržíme tak pouze jiné, stále ale přípustné řešení. Lze tedy předpokládat, že pokud $\theta_{ij} = 0$, pak platí $\beta_{ij} = \gamma_{ij} = 0$, nebo ekvivalentně $\beta_{ij} + \gamma_{ij} - \theta_{ij} = 0$.

Nyní předpokládejme, že $\theta_{ij} = 1$. V takovém případě je nutné, aby $\beta_{ij} = 1$, nebo $\gamma_{ij} = 1$, nebo obojí zároveň. Předpokládejme tedy, že $\beta_{ij} = 1$. V takovém případě je ale opět omezení (3.17) splněno, nezávisle na tom, zda je γ_{ij} rovno nule či jedné. Hodnoty proměnné γ_{ij} zde opět pouze dodávají jiná, stále přípustná řešení. Můžeme tedy předpokládat, že pokud $\theta_{ij} = 1$, pak buď $\beta_{ij} = 1$, nebo $\gamma_{ij} = 1$, ale nikdy obojí zároveň. Ekvivalentně můžeme tuto podmínku opět zapsat ve tvaru rovnosti $\beta_{ij} + \gamma_{ij} - \theta_{ij} = 0$.

Ukázali jsme, že nehledě na volbu θ_{ij}, vždy platí rovnost $\beta_{ij} + \gamma_{ij} - \theta_{ij} = 0$, a tedy můžeme omezení (3.17) nahradit danou rovností. Jestliže tato rovnost platí, můžeme říci, že $\theta_{ij} = \beta_{ij} + \gamma_{ij}$, a proto lze proměnnou θ_{ij} nahradit v omezení (3.16) za součet $\beta_{ij} + \gamma_{ij}$. Toto nahrazení zredukuje celý problém o jednu sadu omezení a vyústí v následující finální zlinearizovaný model.

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\begin{aligned}
\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} &\geq 0, \qquad \forall (i,j) \in A, && (3.18)\\
\alpha_t - \alpha_s &\geq 1,\\
\sum_{(i,j)\in A} r_{ij}\gamma_{ij} &\leq R, && (3.19)\\
\alpha_i &\in \{0,1\}, \qquad \forall i \in N,\\
\beta_{ij}, \gamma_{ij} &\in \{0,1\}, \qquad \forall (i,j) \in A.
\end{aligned}$$

Tento model představuje jinou variantu úlohy o minimálním řezu. Pokud bychom totiž vynechali omezení (3.19) a vypustili bychom γ_{ij} z omezení (3.18), dostali bychom přímo úlohu o minimálním řezu. Minimální řez je v tomto případě tím menší, čím větší je rozpočet útočníka R, protože mu umožní zneprůchodnit větší množství hran. Účelová funkce minimalizuje hodnotu reziduálního minimálního řezu, který se nepodaří útočníkovi porušit. V omezení (3.18) nabývají proměnné α_i hodnoty jedna, platí-li $\alpha_i \in N_t$, v opačném případě proměnná nabývá hodnoty 0. Omezení (3.18) říká, že patří-li hrana (i,j) do minimálního řezu, pak je přerušena útočníkem ($\gamma_{ij} = 1$), nebo ponechána nepoškozena ($\beta_{ij} = 1$). Omezení (3.19) vyjadřuje možnosti útočníka ve formě zdrojů, které může při napadání sítě využít. Vzhledem k odlišné propustnosti hran, si každá jednotlivá hrana vyžaduje jiné množství zdrojů nutné k jejímu přerušení. Tato formulace, z níž vycházejí modifikované modely v kapitole 5, je více rozebrána na konkrétních případech právě v této kapitole, navíc poslední uvedený model je zároveň prvním modelem uvedeným v podkapitole 5.1.

4. Základy stochastického programování

Ještě než přikročíme ke konkrétním deterministickým modelům, je vhodné přehledovou část doplnit základními informacemi o stochastických optimalizačních modelech. V reálném životě často nastává situace, kdy většinu parametrů předem neznáme a máme o nich jen mlhavou představu. Neznáme dopředu množství zboží, které si zákazník objedná, ale víme, že srovnáním všech jeho předchozích objednávek jsme schopni stanovit nějakou dolní a horní mez pro množství objednaného zboží. Použití deterministického modelu může v takovém případě vést k velice chybným výsledkům, viz první kapitola v [4].

Pomocnou ruku nabízí stochastické programování, kde jsou neznámé parametry definovány jako náhodné proměnné. Obecnou formulací úlohy stochastického programování rozumíme, viz [20] a [29].

Definice 4.1. (Úloha stochastického programování)
Nechť trojice $(\Omega, \mathcal{A}, P)$ je pravděpodobností prostor. Zobrazení $\xi : \Omega \to \mathbb{R}^k$ se nazývá náhodný vektor, pokud pro $\forall\, \mathbf{t} \in \mathbb{R}^k$ platí

$$\{\omega : \boldsymbol{\xi}(\omega) \leq \mathbf{t}\} \in \mathcal{A}.$$

Pak lze úlohu stochastického programování zapsat následujícím způsobem

$$\begin{aligned} \min \quad & f(\mathbf{x}, \boldsymbol{\xi}), \\ \text{za podmínek} \quad & \mathbf{g}(\mathbf{x}, \boldsymbol{\xi}) \leq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{0}, \\ & \mathbf{x} \in X, \end{aligned}$$

kde $\boldsymbol{\xi}(\omega) : \Omega \to \mathbb{R}^K$ je náhodný vektor definovaný na pravděpodobnostním prostoru $(\Omega, \mathcal{A}, \mathcal{P})$, $f : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}$, $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$, $\mathbf{h} : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^l$.

Úloha není dobře definována, protože neznáme význam minimalizace $f(\mathbf{x}, \boldsymbol{\xi})$ před pozorováním realizace náhodného vektoru $\boldsymbol{\xi}$ a neznáme ani množinu, na níž se snažíme minimalizovat účelovou funkci $f(\mathbf{x}, \boldsymbol{\xi})$. Obvyklý postup, jak vyřešit takový problém, je zavedení tzv. deterministických reformulací, viz [17]. Zobrazení $\boldsymbol{\xi} : \Omega \to \mathbb{R}^k$, reprezentující náhodnost některých parametrů, indukuje pravděpodobnostní míru $\mathcal{P}$ na prostoru $\mathbb{R}^k$. Dále také předpokládáme, že známe pravděpodobnostní rozdělení vektoru $\boldsymbol{\xi}$, jehož nosič označíme jako $\Xi \subset \Omega$, viz [15]. Předpokládáme, že funkce $f : \mathbb{R}^n \times \Xi \to \mathbb{R}$, $\mathbf{g} : \mathbb{R}^n \times \Xi \to \mathbb{R}^m$ a $\mathbf{h} : \mathbb{R}^n \times \Xi \to \mathbb{R}^l$ jsou měřitelné.

Jaký význam přesně ale má dříve uvedená formulace úlohy stochastického programování? Pozorování realizace náhodného vektoru $\boldsymbol{\xi}^r$ nám umožní v modelu nahradit neznámé parametry a úloha je pak deterministická. Co ale dělat v případě, kdy neznáme realizaci náhodného vektoru $\boldsymbol{\xi}$?

4.1. Deterministické reformulace

Na naší cestě za cílem nalezení řešení úlohy stochastického programování nám pomohou dva základní modelovací přístupy. První přístup nám velí počkat na pozorování realizace

$\boldsymbol{\xi}^r$ náhodného vektoru $\boldsymbol{\xi}$ a poté vyřešit deterministickou úlohu. Takový přístup se nazývá *wait-and-see*, počeštěně „počkej a uvidíš“. Občas ale nastanou situace, kdy nelze s rozhodnutím čekat na realizaci náhodného vektoru $\boldsymbol{\xi}$. V takovém případě je potřeba učinit rozhodnutí vycházející z našich zkušeností, pokud nějaké s daným problémem máme. Takový přístup se nazývá *here-and-now*, počeštěně „tady a teď“. České ekvivalenty názvů přístupu jsme uvedli pouze pro přehled a lepší orientaci. V navazujícím textu budeme uvádět původní anglické termíny.

4.1.1. Wait-and-see přístup

Jak jsme uvedli výše, tento přístup spočívá v tom, že s rozhodnutím $\mathbf{x}$ vyčkáme až na realizaci náhodného vektoru $\boldsymbol{\xi}$. Vzhledem k tomu, že realizace $\boldsymbol{\xi}$ ovlivní rozhodnutí $\mathbf{x}$, je rozhodnutí funkcí náhodného vektoru, tedy $\mathbf{x} = \mathbf{x}(\boldsymbol{\xi})$. Pochopitelně hodnota účelové funkce $f\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big)$ je také náhodná proměnná. Má smysl se tedy ptát na rozdělení $\mathbf{x}(\boldsymbol{\xi})$ a $f\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big)$, které je třeba určit, protože nyní místo klasických proměnných uvažujeme náhodné proměnné.

Deterministická reformulace WS
Mějme úlohu stochastického programování. Její deterministickou reformulaci wait-and-see definujeme jako

$$\begin{aligned} \min \quad & f\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big), \\ \text{za podmínek} \quad & g\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big) \leq \mathbf{0}, \\ & h\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big) = \mathbf{0}, \\ & \mathbf{x}(\boldsymbol{\xi}) \in X. \end{aligned}$$

Zpravidla budeme hodnotu účelové funkce značit z^{WS} a optimální hodnoty rozhodovacích proměnných $\mathbf{x}^{\mathrm{WS}}_{\mathrm{min}}$. Připomeneme, že $z^{\mathrm{WS}} = z^{\mathrm{WS}}(\boldsymbol{\xi})$ a $\mathbf{x}^{\mathrm{WS}}_{\mathrm{min}} = \mathbf{x}^{\mathrm{WS}}_{\mathrm{min}}(\boldsymbol{\xi})$. Pro každou realizaci náhodného vektoru $\boldsymbol{\xi}$ tedy dostáváme jiné hodnoty. Wait-and-see přístup je vhodný pro dlouhodobější rozhodnutí, kde máme informace o budoucím vývoji problému a můžeme na něj reagovat. Pokud takové informace postrádáme a je nezbytné rozhodnout se ihned, lze využít některé z deterministických reformulací here-and-now přístupu.

4.1.2. Here-and-now přístup

Jestliže je nutné učinit rozhodnutí před pozorováním realizace náhodného vektoru $\boldsymbol{\xi}$, pak naše rozhodnutí spadá do kategorie přístupu here-and-now. V takovém případě nejprve učiníme rozhodnutí reprezentované vektorem $\mathbf{x}$ a poté až dochází k realizaci náhodného vektoru $\boldsymbol{\xi}$. Rozhodnutí není závislé na $\boldsymbol{\xi}$ a je stejné pro všechny potenciální možné realizace. Častěji se volí právě tato kategorie přístupů, protože obvykle postrádáme dostatek pozorování, abychom mohli rozhodovat s ohledem na možný budoucí vývoj.

Deterministická reformulace IS
Reformulace IS (z anglického „*individual scenario*“) je založena na myšlence, že v původní úloze stochastického programování nahradíme náhodný vektor $\boldsymbol{\xi}$ nějakou jeho, často typickou, realizací, kterou označíme $\boldsymbol{\xi}^r$. Takovou realizaci $\boldsymbol{\xi}^r$, $r \in \mathcal{S}$, kde $\mathcal{S}$ je množina

indexů možných realizací, pak nazveme *scénář*, nebo také *realizace* a jejich množinu nazveme množinou realizací. Někdy zjednodušeně označíme jako realizaci index r a dále $\mathcal{S}$ zjednodušeně bude množina scénářů. Reformulaci IS lze pak zapsat jako

$$\begin{aligned} \min \quad & f(\mathbf{x}, \boldsymbol{\xi}^r), \\ \text{za podmínek} \quad & \mathbf{g}(\mathbf{x}, \boldsymbol{\xi}^r) \leq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}, \boldsymbol{\xi}^r) = \mathbf{0}, \\ & \mathbf{x} \in X. \end{aligned}$$

Hodnotu účelové funkce pak budeme značit z^{IS} a optimální hodnoty rozhodovacích proměnných $\mathbf{x}^{\mathrm{IS}}_{\min}$. Hodnoty jsou optimální pro vybranou realizaci $\boldsymbol{\xi}^r$ a neposkytují žádnou informaci o hodnotách pro jiné realizace náhodného vektoru. Uvažování konkrétního scénáře má smysl v případě, že máme nablízku někoho s velice dobrou znalostí příslušného problému.

Deterministická reformulace EV

Myšlenka reformulace EV (z anglického „*expected value*“) spočívá v nahrazení realizace náhodného vektoru $\boldsymbol{\xi}$ hodnotami, které ho nějakým způsobem charakterizují, a to vektorem středních hodnot. Reformulaci EV zapíšeme jako

$$\begin{aligned} \min \quad & f(\mathbf{x}, E\boldsymbol{\xi}), \\ \text{za podmínek} \quad & \mathbf{g}(\mathbf{x}, E\boldsymbol{\xi}) \leq \mathbf{0}, \\ & \mathbf{h}(\mathbf{x}, E\boldsymbol{\xi}) = \mathbf{0}, \\ & \mathbf{x} \in X, \end{aligned}$$

kde $E\boldsymbol{\xi}$ reprezentuje střední hodnotu, která nám dovolí zbavit se nejistoty v podobě neznámé realizace $\boldsymbol{\xi}$. Hodnotu účelové funkce budeme značit z^{EV} a optimální hodnoty rozhodujících proměnných $\mathbf{x}^{\mathrm{EV}}_{\min}$. V tomto případě je velice důležité mít kompletní informace o možných realizacích náhodného vektoru.

Deterministická reformulace EO

EO reformulace (z anglického „*expected objective*“ je často užívaná, viz [4] a [17], deterministická reformulace přístupu here-and-now reprezentující střední hodnotu účelové funkce. Reformulaci EO lze zapsat jako

$$\begin{aligned} \min \quad & E\big(f(\mathbf{x}, \boldsymbol{\xi})\big), & \\ \text{za podmínek} \quad & \mathbf{g}(\mathbf{x}, \boldsymbol{\xi}) \leq \mathbf{0}, & s.j. \\ & \mathbf{h}(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{0}, & s.j. \\ & \mathbf{x} \in X, & \end{aligned}$$

kde s.j. značí skoro jistě, viz [17] a [34]. Hodnotu účelové funkce budeme značit z^{EO} a optimální hodnoty rozhodujících proměnných $\mathbf{x}^{\mathrm{EO}}_{\min}$.

Deterministická reformulace MM

Důvodem zavedení reformulace MM („*max-min*“, nebo jako v našem případě „*min-max*“) je nedostatek informací o rozdělení náhodného vektoru $\boldsymbol{\xi}$, viz [29] a [32]. V takovém

případě lze aplikovat defenzivní přístup, jehož smysl spočívá v minimalizaci maximálních ztrát. Reformulaci MM zapíšeme jako

$$\begin{aligned} \min_{\mathbf{x}} \max_{\boldsymbol{\xi}} \quad & f(\mathbf{x}, \boldsymbol{\xi}), \\ \text{za podmínek} \quad & \mathbf{g}(\mathbf{x}, \boldsymbol{\xi}) \leq \mathbf{0}, \quad s.j. \\ & \mathbf{h}(\mathbf{x}, \boldsymbol{\xi}) = \mathbf{0}, \quad s.j. \\ & \mathbf{x} \in X, \end{aligned}$$

kde s.j. značí, stejně jako v předchozím případě, skoro jistě. Hodnotu účelové funkce budeme značit z^{MM} a optimální hodnoty rozhodujících proměnných $\mathbf{x}_{\mathrm{min}}^{\mathrm{MM}}$.

Poznámka:
Výsledky řešení, kdy nejprve rozhodneme bez ohledu na následek, reformulace EV $\mathbf{x}_{\mathrm{min}}^{\mathrm{EV}}$ a EO $\mathbf{x}_{\mathrm{min}}^{\mathrm{EO}}$ se používají pro všechny možné realizace náhodného vektoru $\boldsymbol{\xi}$ a lze je porovnat s přístupem wait-and-see, viz [15].

Poznámka:
Jelikož množinu scénářů budeme uvažovat vždy konečnou, s ohledem na implementaci v programu GAMS, střední hodnota $E\boldsymbol{\xi}$ je pak určena

$$E\boldsymbol{\xi} = \sum_{r \in \mathcal{S}} p_r \boldsymbol{\xi}^r \quad \text{a} \quad Ef(\mathbf{x}, \boldsymbol{\xi}) = \sum_{r \in \mathcal{S}} p_r f(\mathbf{x}, \boldsymbol{\xi}^r),$$

kde p_r představuje pravděpodobnost realizace náhodného vektoru $\boldsymbol{\xi}^r$, více podrobností o scénářových úlohách v podkapitole 4.3.

4.2. Dvoustupňové stochastické programování

Až doposud jsme se věnovali úlohám stochastického programování, kde bylo třeba učinit jedno rozhodnutí s ohledem na náš cíl. V této části se budeme věnovat dvoustupňovým stochastickým úlohám. Takové úlohy vyžadují dvě rozhodnutí, která udělá takzvaný rozhodovatel. Kdy je ale udělat? Dělají se zároveň?

První rozhodnutí dělá rozhodovatel na počátku řešení problému. V úvodní chvíli rozhodovatel postrádá jakoukoli informaci o budoucím vývoji situace, respektive informaci o realizaci náhodných proměnných. Takové rozhodnutí nazveme *rozhodnutí v prvním stupni*, a dobu, kdy rozhodnutí učiníme, pak *první stupeň*.

Druhé rozhodnutí v případě úlohy dvoustupňového stochastického programování konáme až po realizaci náhodných proměnných. Situace nám dovolí reagovat na důsledky našeho prvního rozhodnutí. Rozhodnutí učiněné po realizaci náhodných proměnných nazveme *rozhodnutí ve druhém stupni*. Moment, kdy učiníme rozhodnutí, nazveme *druhý stupeň*.

Jediné, co z hlediska času uvažujeme, je pořadí akcí. Nejprve rozhodneme v prvním stupni, pak se realizují náhodné proměnné, pak rozhodneme ve druhém stupni. Čas mezi rozhodnutími není obecně pevně dán. V podkapitole 3.3 jsme uvažovali dvouúrovňové rozhodování, kdy zájmy více (v tomto případě dvou) rozhodovatelů mohou být různé,

zatímco zde uvažujeme rozhodování jednoho rozhodovatele ve dvou časových momentech, tj. stupních.

Podotýkáme, že tento typ úloh je vlastně kombinací here-and-now a wait-and-see přístupu. Here-and-now přístup je skryt v první úrovni, kdy je třeba rozhodnout, aniž bychom měli nějakou představu o budoucnosti. Wait-and-see přístup skrývá druhá úroveň, kdy jsme počkali na výsledek našeho rozhodnutí a realizace náhodných proměnných $\boldsymbol{\xi}$.

Rozhodnutí v prvním stupni tedy budeme značit $\mathbf{x}$ a rozhodnutí ve druhém stupni pak $\mathbf{y}(\boldsymbol{\xi})$, protože závisí na realizaci náhodných parametrů. Rozhodnutí $\mathbf{y}(\boldsymbol{\xi})$ závisí obecně i na rozhodnutí $\mathbf{x}$ z prvního stupně, ale konvence značení (viz [29] a [30]) je tuto závislost nezdůrazňovat zápisem $\mathbf{y}(\boldsymbol{\xi}, \mathbf{x})$.

Lineární dvoustupňové úlohy s kompenzací

Důležitou úlohou stochastického programování jsou lineární dvoustupňové úlohy s kompenzací, viz [4], [17]. Důležitou myšlenkou lineárních dvoustupňových úloh s kompenzací je, že na počátku rozhodovatel vykoná rozhodnutí $\mathbf{x}$, aniž by věděl, co bude následovat, tj. jaká bude realizace náhodných parametrů $\boldsymbol{\xi}$. Úloha pak pokračuje druhým stupněm, kdy už rozhodovatel zná realizaci náhodných proměnných $\boldsymbol{\xi}$ a může pokročit k rozhodnutí $\mathbf{y}$. Cílem v těchto úlohách je opravit, tj. kompenzovat, výsledek prvního rozhodnutí $\mathbf{x}$, kvůli možnému nevýhodnému projevu náhodných proměnných $\boldsymbol{\xi}$, nebo napravit případnou neřešitelnost úlohy způsobenou realizací náhodné proměnné. Druhý stupeň, v němž se opravuje naše předchozí rozhodnutí, nazýváme *kompenzace*. Je zřejmé, že kompenzace $\mathbf{y}(\boldsymbol{\xi})$ zvyšuje hodnotu účelové funkce. Pro kompenzací se rozhodneme tedy pouze v případě, že skutečně bude mít za následek pozitivní vliv na řešení. Pokud není třeba využít kompenzaci, pro druhou úroveň platí $\mathbf{y}(\boldsymbol{\xi}) = \mathbf{0}$. Taková úloha se nazývá stochastická úloha s kompenzací.

Nyní uvedeme lineární dvoustupňovou úlohu stochastického programování s pevnou kompenzací. Ještě předtím je však třeba poukázat na fakt, že úloha druhého stupně je deterministická, protože už známe realizaci náhodných proměnných $\boldsymbol{\xi}$. Důležitou vlastností ve dvoustupňových úlohách a dalších výše zmíněných je požadavek na to, aby rozhodnutí v první úrovni byla *neanticipativní*. To znamená, že chceme, aby rozhodovatel učinil rozhodnutí v první úrovni ještě před tím než pozná, jakých hodnot budou nabývat parametry $\boldsymbol{\xi}$, které by jeho rozhodnutí mohly ovlivnit. To znamená, že rozhodnutí v první úrovni $\mathbf{x}$ je nezávislé na čemkoli, co se stane v budoucnosti. Lineární dvoustupňovou úlohou s kompenzací rozumíme

$$\min_{\mathbf{x}}\Big\{\mathbf{c}^\top\mathbf{x} + E\Big[\min_{\mathbf{y}(\boldsymbol{\xi})}\big\{\mathbf{q}^\top(\boldsymbol{\xi})\mathbf{y}(\boldsymbol{\xi})\big|\mathbf{T}(\boldsymbol{\xi})\mathbf{x} + \mathbf{W}\mathbf{y}(\boldsymbol{\xi}) = \mathbf{h}(\boldsymbol{\xi}), \mathbf{y}(\boldsymbol{\xi}) \geq \mathbf{0}\big\}\Big]\Big|\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0\Big\}.$$

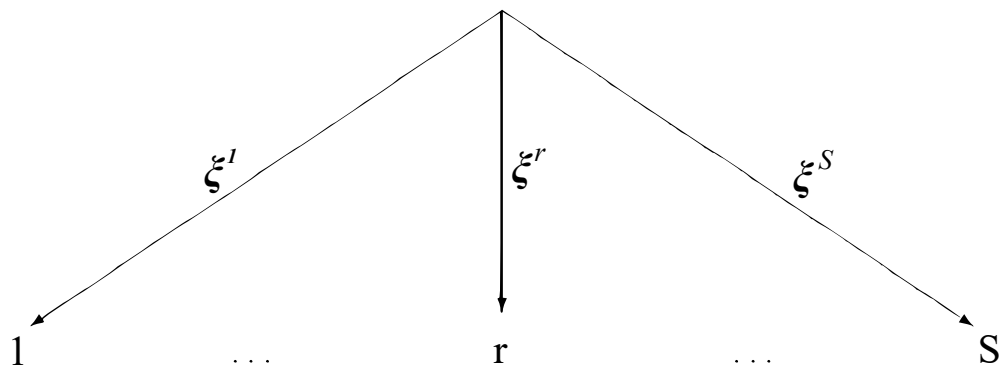
Tak jako vektor $\mathbf{c}$ reprezentuje ceny pro rozhodnutí $\mathbf{x}$ v prvním stupni, podobně vektor $\mathbf{q}(\boldsymbol{\xi})$ reprezentuje cenu pro kompenzační rozhodnutí $\mathbf{y}(\boldsymbol{\xi})$ ve druhém stupni. Omezení $\mathbf{A}\mathbf{x} = \mathbf{b}$ se vzhledem k absenci $\mathbf{y}$ vztahuje k rozhodnutí v prvním stupni. Pro danou realizaci náhodného vektoru $\boldsymbol{\xi}$ získáme data pro druhý stupeň $\mathbf{q}(\boldsymbol{\xi})$, $\mathbf{T}(\boldsymbol{\xi})$ a $\mathbf{h}(\boldsymbol{\xi})$. Tuto stochastickou dvoustupňovou úlohu nazveme úlohou s pevnou kompenzací, jelikož $\mathbf{W}$ nezávisí na realizaci $\boldsymbol{\xi}$. Omezení $\mathbf{T}(\boldsymbol{\xi})\mathbf{x} + \mathbf{W}\mathbf{y}(\boldsymbol{\xi}) = \mathbf{h}(\boldsymbol{\xi})$ a $\mathbf{y}(\boldsymbol{\xi}) \geq \mathbf{0}$ tedy propojují první a druhý stupeň. Dvoustupňové úlohy lze uvažovat i nelineární, dále lze uvažovat i vícestupňové úlohy, více například v [4].

Ukázali jsem základní model dvoustupňové lineární stochastické úlohy s kompenzací. Existují taktéž úlohy, kde rozhodnutí ve druhém stupni není pouhou kompenzací, ale je vyžadováno samotnými omezeními vztahujícími se ke druhému stupni. V takovém případě je nutné rozhodnutí ve druhém stupni udělat, přestože kompenzace nemusí být potřeba. Takové úlohy se někdy obecně nazývají dvoustupňové stochastické úlohy a slovo kompenzace se explicitně neuvádí, viz [29].

4.3. Scénářové úlohy stochastického programování

Další možný přístup k stochastických úloh, nejen v logistice, nabízí scénářové úlohy stochastického programování. Tento přístup může být velice užitečný a praktický ve chvíli, kdy mají náhodné parametry $\boldsymbol{\xi}$ konečně diskrétní rozdělení, tj. $|\Xi| < \infty$.

Myšlenka scénářových úloh spočívá v tom, že nejistoty a náhodnost proměnných můžeme modelovat pomocí *scénářů*. Scénář r představuje realizaci vektoru náhodných proměnných $\boldsymbol{\xi}^r$. Množinu scénářů označíme jako $\mathcal{S} = \{1, ..., S\}$, kde S je počet scénářů, a symbolem p_r budeme označovat pravděpodobnost, že scénář $r \in \mathcal{S}$ nastane. Scénáře lze pak zobrazit scénářovým stromem, viz obrázek 4.1 převzatý z [29].



Obrázek 4.1: Scénářový strom pro dvoustupňovou úlohu.

Přístupy k práci se scénáři vzhledem k jejich množství jsou dva. První možností je, že pokud je možných scénářů méně, lze úlohu spočítat pro všechny scénáře. Jelikož s množstvím scénářů stoupá i výpočetní náročnost, pak se v případě skutečně velkého počtu scénářů používají metody redukce počtu scénářů, viz [4].

Scénářům jsou přiřazeny pravděpodobnosti podle toho, jestli nastávají častěji nebo představují reálnější realizaci vektoru náhodných proměnných $\boldsymbol{\xi}$. Pro pravděpodobnosti platí, že $p_r = P(\boldsymbol{\xi} = \boldsymbol{\xi}^r) \geq 0$ a $\sum_{r \in \mathcal{S}} p_r = 1$.

Pro jednoduchost budeme v následující části značit $\mathbf{y}_r = \mathbf{y}(\boldsymbol{\xi}^r)$, $\mathbf{q}_r = \mathbf{q}(\boldsymbol{\xi}^r)$, $\mathbf{W}_r = \mathbf{W}(\boldsymbol{\xi}^r)$, $\mathbf{T}_r = \mathbf{T}(\boldsymbol{\xi}^r)$, $\mathbf{h}_r = \mathbf{h}(\boldsymbol{\xi}^r)$, $\forall r \in \mathcal{S}$. Scénářovou dvoustupňovou stochastickou lineární úlohu lze zapsat takto

$$\min_{\mathbf{x}} \mathbf{c}^\top \mathbf{x} + \sum_{r \in \mathcal{S}} p_r Q(\mathbf{x}, \boldsymbol{\xi}^r),$$

$$\begin{aligned} \text{za podmínek} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{4.1}$$

kde

$$Q(\mathbf{x}, \boldsymbol{\xi}^r) = \min_{\mathbf{y}_r} \{\mathbf{q}_r^\top \mathbf{y}_r\}$$

$$\begin{aligned} \text{za podmínek} \quad \mathbf{T}_r\mathbf{x} + \mathbf{W}_r\mathbf{y}_r &= \mathbf{h}_r, \quad r \in \mathcal{S}, \\ \mathbf{y}_r &\geq \mathbf{0}, \quad r \in \mathcal{S}. \end{aligned} \tag{4.2}$$

Výše zapsanou úlohu přepíšeme častěji užívaným způsobem vhodným pro implementaci v GAMSu, viz [20]:

$$\min_{\mathbf{x},\mathbf{y}_r} \mathbf{c}^\top \mathbf{x} + \sum_{r \in \mathcal{S}} p_r \mathbf{q}_r^\top \mathbf{y}_r$$

za podmínek

$$\begin{aligned} \mathbf{A}\mathbf{x} & & & & = \mathbf{b}, \\ \mathbf{T}_1\mathbf{x} + \mathbf{W}_1\mathbf{y}_1 & & & & = \mathbf{h}_1, \\ \mathbf{T}_2\mathbf{x} & + \mathbf{W}_2\mathbf{y}_2 & & & = \mathbf{h}_2, \\ \vdots & & \ddots & & \vdots \\ \mathbf{T}_S\mathbf{x} & & & + \mathbf{W}_S\mathbf{y}_S & = \mathbf{h}_S, \end{aligned}$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{y}_1, ..., \mathbf{y}_S \geq \mathbf{0}.$$

Je potřeba si uvědomit, že se jedná o zápis pomocí matic a vektorů, a tedy s rostoucím množstvím scénářů velice rychle roste výpočetní náročnost řešení programu.

5. Modely napadení sítě

Ze všeho nejdříve by bylo vhodné připomenout, co vlastně rozumíme pojmem *napadení sítě*. Napadením sítě se rozumí přerušení či kontrola uzlů, hran nebo kombinace obojího zároveň. V následujících modelech se jedná o přerušování, a to úplné i částečné. Útočník, který napadá síť, tak svým útokem zamezí průchodu přes danou část sítě. Jednoduchou analogii nám poskytne například policejní zátaras na silnici, nebo letištní kontrola. Cílem útočníka je zpravidla minimalizace toku v síti nebo zachycení nevyžádaných elementů v toku s jistou pravděpodobností, jak uvádí [25]. Budeme využívat značení převzatého především z [18] a [33], [37].

Pro výpočty budeme využívat software GAMS (*The General Algebraic Modeling System*). Program byl vybrán na základě dobré dostupnosti, nenáročné syntaxe a uživatelské přívětivosti. GAMS je optimalizační software využívající vlastní programovací jazyk a implementované řešiče, které při výpočtech využíváme. Mezi výhody GAMSu patří snadný zápis modelu včetně omezení, možnost propojení s MS Excel pro import a export dat a zároveň možnost exportovat data do textového souboru, viz [5], [11].

My v této kapitole využíváme řešič CPLEX, k jehož přednostem patří stabilita, značná rychlost a schopnost řešení úloh lineárního programování bez dalšího zásahu uživatele. Řešič k výpočtům využívá simplexovou metodu, případně duální simplexovou metodu, viz [5]. Pro import dat z MS Excel lze využít v GAMSu vestavěného nástroje GDXXRW. Přesný návod na to, jak data importovat, je uveden v [5] nebo [11].

V této části je kladen důraz na popis jednotlivých modelů napadení z pohledu lineárního programování, viz kapitola 3. Řešení úlohy má dvě na sebe navazující úrovně. V první úrovni je síť napadena útočníkem. Druhou úroveň představuje úloha o maximálním toku v porušené síti. Různé způsoby napadání sítě jsou podrobně popsány v následujících částech. Uvedena je pouze první úroveň přeformulované dvouúrovňové úlohy, protože druhá úroveň je pro všechny modely stejná, viz úloha o maximálním toku v síti (kapitola 2):

$$\max x_{ts},$$

za podmínek

$$\begin{aligned}
\sum_{j\in N'} x_{sj} - \sum_{i\in N'} x_{is} - x_{ts} &\leq 0, \\
\sum_{j\in N} x_{nj} - \sum_{i\in N} x_{in} &\leq 0, \qquad \forall n \in N', \\
\sum_{j\in N'} x_{tj} - \sum_{i\in N'} x_{it} + x_{ts} &\leq 0, \\
0 \leq x_{ij} &\leq u_{ij}(1-\gamma_{ij}), \qquad \forall (i,j) \in A.
\end{aligned}$$

Cílem je uvést možnosti formulace problémů s pomocí různých omezení v závislosti na aplikaci nebo na užitých datech. První tři následující podkapitoly uvádějí modely inspirované [37] včetně vlastní softwarové implementace, následující tři modely navazují na [18] a jsou rovněž implementovány v GAMSu.

5.1. Model pro přerušování hran

První model je určen pro přerušování hran. Máme dánu síť dle definice 2.21 z kapitoly 2 a matici konstantních kapacit orientovaných hran $\mathbf{U} = (u_{ij})$. Cílem útočníka je minimalizovat tok v síti, kterou svým napadením poničí. Možný tok v síti zde reprezentuje hodnota nepřerušeného minimálního řezu, viz podkapitoly 2.2 a 3.2. Cena za napadení orientované hrany (i, j) mezi uzly $i, j \in N$ je definována maticí $\mathbf{C} = (c_{ij})$, jejíž prvky jsou konstantní. V této úloze předpokládáme jediný počáteční uzel, značíme $s \in N$, a jediný cílový uzel, značíme $t \in N$. Jediným omezením útočníka je jeho rozpočet označený parametrem R. Úlohu lze formulovat následujícím způsobem, viz [37]:

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} \geq 0, \qquad \forall (i,j) \in A, \tag{5.1}$$

$$\alpha_t - \alpha_s \geq 1, \tag{5.2}$$

$$\sum_{(i,j)\in A} c_{ij}\gamma_{ij} \leq R, \tag{5.3}$$

$$\alpha_i \in \{0,1\}, \qquad \forall i \in N,$$

$$\beta_{ij}, \gamma_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A,$$

kde

$$\alpha_i = \begin{cases} 1 & \text{pokud se uzel nachází na straně řezu patřícímu cílovému uzlu } t, \\ 0 & \text{pokud se uzel nachází na straně řezu patřícímu počátečnímu uzlu } s, \end{cases}$$

$$\beta_{ij} = \begin{cases} 1 & \text{pokud hrana } (i,j) \text{ náleží do minimálního řezu, ale není přerušena,} \\ 0 & \text{jinak,} \end{cases}$$

$$\gamma_{ij} = \begin{cases} 1 & \text{pokud náleží hrana } (i,j) \text{ do minimálního řezu a je přerušena,} \\ 0 & \text{jinak.} \end{cases}$$

Podle definice proměnných β_{ij} a γ_{ij} mohou pro hrany (i, j) mezi uzly $i, j \in N$ nastat tři možnosti. V případě, že hrana (i, j) nenáleží do minimálního řezu, jsou β_{ij} i γ_{ij} nulové. Pokud hrana (i, j) náleží do minimálního řezu, pak je v případě přerušení $\beta_{ij} = 0$ a $\gamma_{ij} = 1$. Pokud hrana (i, j) není přerušena a patří do minimálního řezu, tak je $\beta_{ij} = 1$ a $\gamma_{ij} = 0$. Účelová funkce tedy v kontextu Ford-Fulkersonovy věty (viz věta 2.33) minimalizuje tok v síti přes nepřerušené hrany minimálního řezu.

Omezení (5.1) je splněno pro hrany nenáležící do minimálního řezu, neboť hodnoty α_i a α_j jsou si rovny a zároveň β_{ij} a γ_{ij} jsou obě nulové. V případě hran náležících do minimálního řezu je rozdíl hodnot α_i a α_j roven mínus jedné. Omezení vynucuje, vzhledem k definici proměnných, aby právě jedna hodnota, buď β_{ij}, nebo γ_{ij}, byla rovna jedné a omezení tak bylo splněno i pro hrany patřící do minimálního řezu. Omezení (5.2)

dává návod na to, ve kterém směru probíhá tok, a v případě orientovaného grafu na to, který směr hran přerušovat. Omezení (5.3) dává do souvislosti součet cen c_{ij} přerušených hran, které nesmí překročit útočníkův rozpočet R.

Nyní uveďme konkrétní příklad pomocí zápisu v jazyce GAMS, viz [11].

```
---------------------------- BDM.gms model ----------------------------
Sets
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/;
Alias (i,j);
```

Nejprve musíme v programu zavést dále používané množiny. V našem případě v programu nejprve zavedeme množinu všech uzlů, kterou značíme `i` a její podmnožinu `n(i)` označující vnitřní uzly v síti. Příkazem `alias` vytvoříme množinu `j`, která obsahuje stejné prvky jako množina `i`. Označení množin v programu je trochu pozměněno oproti grafickému zápisu, viz obrázek 5.1, protože program GAMS neumožňuje pojmenování množin s využitím apostrof.

```
Table u(i,j)
    s     n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0     1000  1000  1000  0     0     0     0     0     0     0
n1  0     0     13    0     15    15    0     0     0     0     0
n2  0     13    0     5     13    6     12    0     0     0     0
n3  0     0     5     0     0     13    12    0     0     0     0
n4  0     15    13    0     0     5     0     13    10    0     0
n5  0     15    6     13    5     0     11    6     14    15    0
n6  0     0     12    12    0     11    0     0     2     9     0
n7  0     0     0     0     13    6     0     0     18    0     1000
n8  0     0     0     0     10    14    2     18    0     2     1000
n9  0     0     0     0     0     15    9     0     2     0     1000
t   1000  0     0     0     0     0     0     0     0     0     0;
Table c(i,j)
    s     n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0     1000  1000  1000  0     0     0     0     0     0     0
n1  0     0     1     0     1     1     0     0     0     0     0
n2  0     1     0     1     1     1     1     0     0     0     0
n3  0     0     1     0     0     1     1     0     0     0     0
n4  0     1     1     0     0     1     0     1     1     0     0
n5  0     1     1     1     1     0     1     1     1     1     0
n6  0     0     1     1     0     1     0     0     1     1     0
n7  0     0     0     0     1     1     0     0     1     0     1000
n8  0     0     0     0     1     1     1     1     0     1     1000
n9  0     0     0     0     0     1     1     0     1     0     1000
t   1000  0     0     0     0     0     0     0     0     0     0;

Parameter
budget / 3 /;
```

Zde jsme zapsali vstupní testovací data pro uvažovanou síť, viz obrázek 5.1. Tabulka `u(i,j)` reprezentuje kapacitní matici **U**, obdobně tabulka `c(i,j)` představuje matici cen za napadení hran **C**. Z úsporných důvodů zapíšeme proměnné alternativním zápisem oproti vlastní implementaci v programu GAMS.

```
Binary variable
alpha(i), gamma(i,j), beta(i,j);

Positive variable
x(i,j);

Variable
z1, z2, iflow(i,j), util(i,j);
```

V další části kódu v programu GAMS definujeme proměnné, které budeme při výpočtu potřebovat. Proměnné `alpha` a `beta`, `gamma` mají totožný význam jako proměnné v omezeních (5.1) - (5.3) dříve zapsaného modelu. Proměnná `x` vyjadřující tok na hranách slouží pro výpočet maximálního toku v napadené síti. Model pro maximální tok v síti jsme uvedli v 3.2. Zde ani v dalších modelech napadení sítě ho opakovat nebudeme, protože zůstává beze změny. Uvedeme pouze jeho zápis v programu GAMS. Do proměnné `z1` se bude zapisovat hodnota účelové funkce výše uvedeného modelu. Proměnná `z2` pak slouží jako proměnná vyjadřující maximální tok v síti po jejím napadení. Proměnná `iflow (i,j)` slouží k vyjádření přerušeného toku na jednotlivých hranách v úloze napadení sítě. Proměnná `util(i,j)` se využívá k zápisu kapacitní matice **U** napadené sítě. Proměnné `iflow (i,j)` a `util(i,j)` se v omezeních modelu napadení sítě nevyskytují, ale v programu GAMS jsou využity pro přenos dat o napadené síti, respektive o výsledku napadení, do navazující úlohy o maximálním toku. Dále je potřeba zapsat omezení modelu, a to následujícím způsobem. Každé omezení je třeba v programu GAMS označit unikátním názvem, protože jednotlivá omezení lze kombinovat i pro více modelů.

```
Equations  obj1        ucelova funkce pro utocnika
           bal1(i,j)   omezeni zachovavajici smer toku v siti
           stc1        omezeni pro pocatecni uzel
           stc2        omezeni pro cilovy uzel
           bud         utocnikuv rozpocet
           iflw(i,j)   preruseny tok
           iutil(i,j)  kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0).. alpha(i) - alpha(j) + beta(i,j) + gamma(i,j) =g= 0;
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud.. sum((i,j), c(i,j) * gamma(i,j)) =l= budget;
iflw(i,j).. iflow(i,j) =e= u(i,j) * gamma(i,j);
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);
```

Výše zmíněné názvy rovnic a nerovnic dále provážeme s konkrétními rovnicemi a nerovnicemi, které patří do modelu pro napadání hran. Popis názvů rovnic je čistě pracovní, aby bylo snazší asociovat si názvy rovnic s konkrétními rovnicemi v modelu.

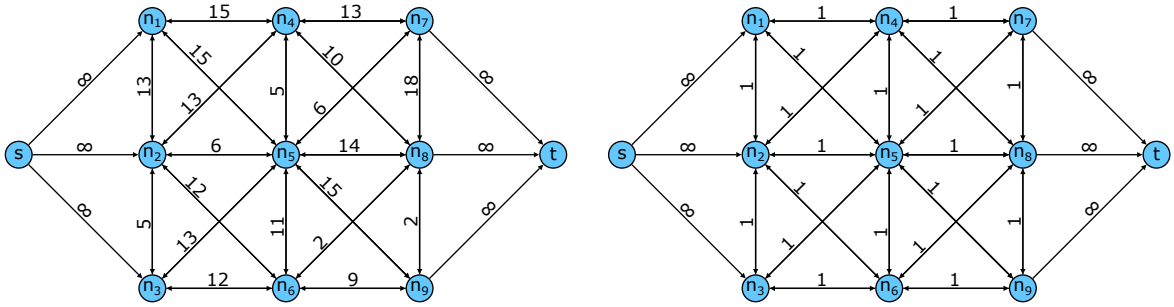
Rovnici od nerovnice lze v GAMSu poznat tak, že rovnice obsahuje `=e=` (z angl. *equal*), zatímco nerovnice obsahují buď `=l=` (z angl. *less or equal*) vyjadřující „menší nebo rovno“, nebo `=g=` (z angl. *greater or equal*) vyjadřující „větší nebo rovno“. První rovnice `obj1` představuje účelovou funkci, o jejíž minimalizaci se snažíme. Následující nerovnice `bal1(i,j)` vyjadřuje omezení (5.1). Omezení (5.2) je zapsáno ve tvaru dvou rovností `stc1('s')` a `stc2('t')`. Rovnice `bud` vyjadřuje omezení rozpočtu útočníka, viz omezení (5.3). Rovnice `iflw(i,j)` vyjadřuje přerušený tok na hranách a kapacity nezničených hran, které lze využít v navazující úloze o maximálním toku, jejíž model následuje, získáme z rovnice `iutil(i,j)`. Připomeňme, že rovnice `iflw(i,j)` a `iutil(i,j)` nepatří do modelu napadení sítě a slouží jen k přenesení dat mezi modely.

```
Equations  obj2     ucelova funkce pro uzivatele
           bal2(n)  omezeni zachovavajici tok v uzlu
           src2     omezeni pro tok z pocatecni uzel
           ter2     omezeni pro tok do ciloveho uzlu
           xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s') =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s') =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);
```

Využíváme model pro maximální tok v síti, který jsme představili v podkapitole 3.2. Zde je vidět, že zápis modelu pomocí nerovnic, který jsme využili kvůli odvození duální úlohy, nic nemění na charakteru úlohy, přestože jsme v definici 2.22 toku v síti požadovali platnost Kirchhoffových zákonů. Rovnice `obj2` vyjadřuje účelovou funkci, maximalizaci toku přes návratovou hranu. Omezení (3.2) je zapsáno jako omezení `bal2(n)`. Omezení (3.1) pro počáteční uzel `s` a omezení (3.3) pro cílový uzel `t` jsou zapsána nerovnicemi `src2` a `ter2`. Nerovnice `xc(i,j)` představuje kapacitní omezení (3.4) pro data získaná z předchozího modelu. Výše zapsaná omezení pro oba modely je ještě třeba svázat pomocí příkazů `model` a `solve`. Do prvního příkazu mezi lomené čáry zapisujeme názvy rovnic a nerovnic, které se k danému modelu vztahují a které budou využity při optimalizaci účelové funkce. Následně druhý příkaz modely vyřeší. V tomto případě je velice důležité pořadí zápisu, protože v prvním modelu vypočítáme data, která používáme následně v druhém modelu.

```
model BDMattack /obj1, bal1, stc1, stc2, bud, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, gamma.l, iflow.l, x.l, z2.l;
```

Následně pomocí příkazu `display` umožníme výpis námi požadovaných výsledků, případně i původních parametrů do výstupního souboru .lst v GAMSu. Uvažujeme následující síť, která je grafickým zobrazením dříve uvedených dat.



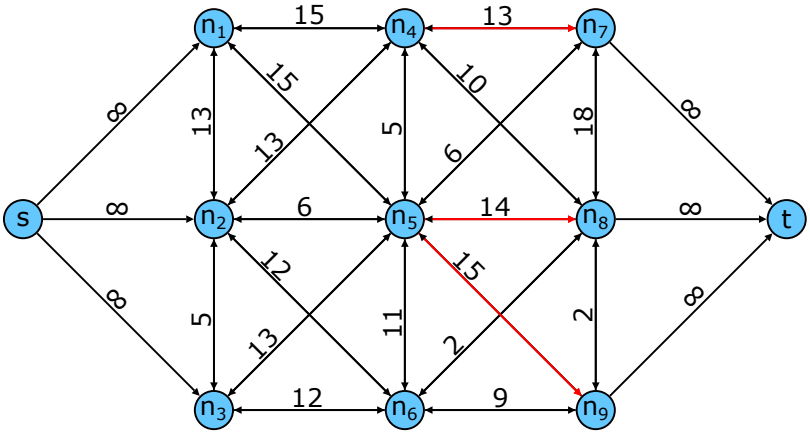
(a) Kapacity hran u_{ij} (b) Ceny za přerušení hran c_{ij}

Obrázek 5.1: Zadání modelové úlohy.

Čísla na hranách v obrázku 5.1a reprezentují kapacity hran a v obrázku 5.1b ceny za přerušení příslušných hran. Předpokládejme, že cena za přerušení všech hran vystupujících z počátečního uzlu $s \in N$ a vstupujících do cílového uzlu $t \in N$ je nekonečná stejně jako jejich kapacity. „Nekonečnou“ kapacitu hran i cenu za přerušení jsme v GAMSu modelovali čísly mnohonásobně převyšujícími možné využití kapacit i dostupný rozpočet, viz dříve uvedené tabulky vstupů.

Síť tedy obsahuje množinu uzlů $N = \{s, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, t\}$, kterou jsme dříve kvůli potřebám programu označili i. V případě značení proměnných pro hrany (i, j) mezi uzly $i, j \in N$ budeme nadále užívat zjednodušené značení, tj. například γ_{12} je hodnota proměnné γ_{ij} pro hranu (n_1, n_2). Hodnoty kapacit i cen za přerušení jsou záměrně uvedeny grafickou formou, protože zápis matic 11×11 není příliš ilustrativní.

Řešením zadané modelové úlohy pro dříve uvedený model je následující síť na obrázku 5.2, v níž jsou přerušené orientované hrany označeny červeně.

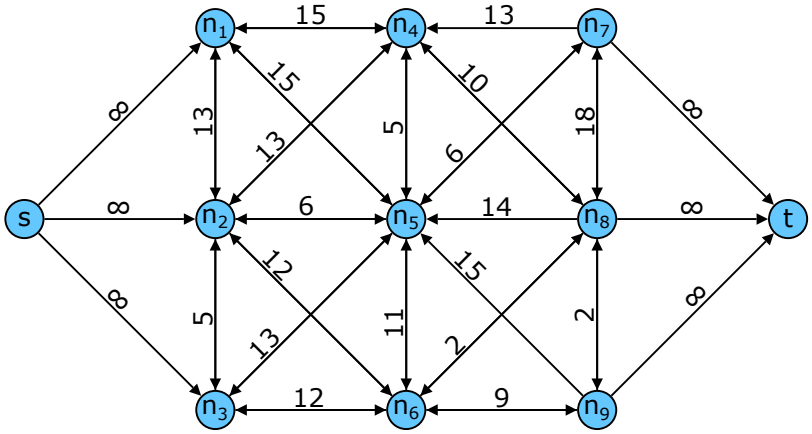


Obrázek 5.2: Řešení modelové úlohy pro model 5.1.

Při podrobném pohledu vidíme, že orientované hrany jsou přerušovány „jednosměrně“. Přerušený směr je zvýrazněn červenou šipkou. Minimální řez tvoří hrany vycházející z podmnožiny uzlů $\{n_4, n_5, n_6\}$ a vstupující do podmnožiny uzlů $\{n_7, n_8, n_9\}$. Optimální hodnoty jednotlivých proměnných v úloze napadení sítě jsou následující:

```
---------------------------- output BDM.gms ----------------------------
---- 120 VARIABLE alpha.L
n7 1.000,     n8 1.000,     n9 1.000,     t 1.000

---- 120 VARIABLE beta.L
           n7          n8          n9
n4                  1.000
n5      1.000
n6                  1.000       1.000


---- 120 VARIABLE gamma.L
           n7          n8          n9
n4      1.000
n5                  1.000       1.000

---- 120 VARIABLE iflow.L
            n7          n8          n9
n4      13.000
n5                   14.000      15.000

---- 120 VARIABLE z2.L     =     27.000
```

V následujícím obrázku 5.3, představujícím síť po napadení útočníkem, jsou šipky orientovaných hran záměrně zvětšeny, aby bylo zřejmé, jak se útok projeví. Druhou úroveň o maximálním toku pak představuje tok v napadené síti.



Obrázek 5.3: Síť po napadení útočníkem 5.1.

Optimálním řešením je tedy přerušení hran $(n_4, n_7), (n_5, n_8)$ a (n_5, n_9) snižující maximální možný tok v síti z 69 na 27. Hodnota původního maximálního toku 69 byla vypočtena při nastavení rozpočtu útočníka na nulu. Útočník neměl prostředky pro napadení sítě, a tedy ji nijak nepoškodil. Na obrázku 5.2 lze vidět, že pro úplné přerušení toku v síti by útočník potřeboval rozpočet $R = 7$, aby přerušil navíc zbývající hrany mezi skupinami uzlů (n_4, n_5, n_6) a (n_7, n_8, n_9). Řešení problému a popsaný model včetně komentářů lze nalézt v přiloženém souboru *BDM.gms*. Hrany vystupující z první skupiny a vstupující do druhé skupiny uzlů pak definují minimální řez v dané síti.

5.2. Model pro přerušování hran s více zdroji

Následující model rozšiřuje a navazuje na předchozí model. Předpokládáme, že pro přerušení hran je třeba více zdrojů, například policistů a policejních aut, patřících do množiny K, jejíž kardinalitu, tedy počet prvků, označíme $|K|$. Změnou je také možnost hranu zničit zcela nebo částečně. Model zapíšeme následujícím způsobem:

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\alpha_i - \alpha_j + \beta_{ij} + \sum_k \frac{\gamma_{ijk}}{|K|} \geq 0, \qquad \forall (i,j) \in A, \tag{5.4}$$

$$\alpha_t = 1, \tag{5.5}$$

$$\alpha_s = 0, \tag{5.6}$$

$$\sum_{(i,j)\in A} c_{ijk}\gamma_{ijk} \leq R_k, \qquad \forall k \in K, \tag{5.7}$$

$$\alpha_i \in \{0,1\}, \qquad \forall i \in N,$$

$$0 \leq \beta_{ij} \leq 1, \qquad \forall (i,j) \in A,$$

$$0 \leq \gamma_{ijk} \leq 1, \qquad \forall (i,j) \in A, \forall k \in K.$$

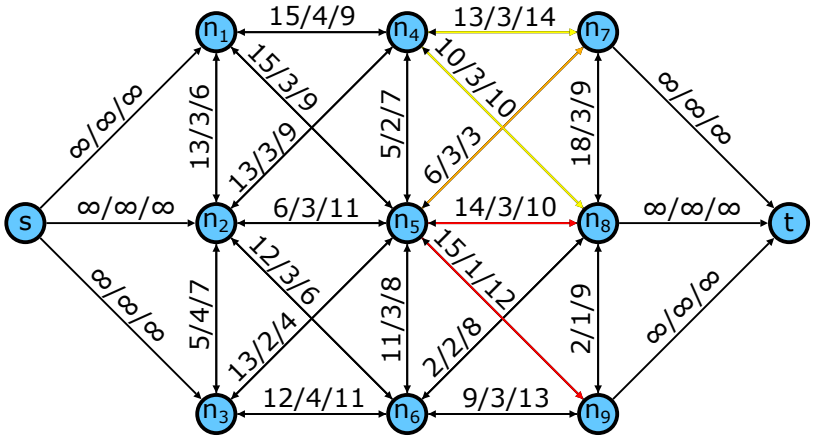
Význam všech proměnných je stejný jako v předchozím modelu z podkapitoly 5.1 s výjimkou proměnné γ_{ijk}. Potřebu pro přerušení hrany pomocí více zdrojů zachycuje omezení (5.4), které každé surovině přiřazuje stejnou váhu při přerušování hrany a které je analogické s omezením (5.1). Zlomek v omezení je navíc záměrně nastaven tak, aby nabýval hodnot mezi 0 a 1 a s tím i hodnota proměnné β_{ij} nabývá hodnot mezi 0 a 1. Představme si situaci, kdy je potřeba pro přerušení hrany policejní auto (s policistou uvnitř) a navíc další dva policisté. Omezení pak říká, že pokud máme dva policisty bez policejního auta, přeruší hranu pouze z jedné poloviny. Pokud máme navíc i policejní auto, dojde k přerušení hrany zcela. Všechny zdroje jsou stejně významné a nezávisle na množství mohou přerušit část kapacity rovnu $\frac{1}{|K|}$. Pokud bychom chtěli zachytit, že některé zdroje jsou významnější než jiné, vyměnili bychom dělení kardinalitou množiny K za součin s námi definovaným váhovým vektorem, jehož součet všech složek by byl opět roven jedné. Rovnosti (5.5) a (5.6) představují ekvivalentně zapsané omezení (5.2) z předchozího modelu. Tento zápis využijeme v následující podkapitole. Omezení (5.7) rozšiřuje omezení (5.3) z předchozího modelu tak, že platí pro každou potřebnou surovinu zvlášť.

Pro úplnost uvedeme také zápis omezení (5.4), které se v modelu v podkapitole 5.1 nevyskytovalo, v programu GAMS.

```
bal1(i,j)$(u(i,j)>0).. alpha(i) - alpha(j) + beta(i,j) +
sum(k,gamma(i,j,k)/card(k)) =g= 0;
```

Uvažujme následující síť, kde čísla na hranách postupně vyjadřují kapacitu příslušné hrany, množství zdroje 1 a množství zdroje 2 potřebné pro přerušení hrany. Rozpočet útočníka reprezentuje vektor $R = (10, 25)$. Opět předpokládáme, že hrany vycházející z počátečního a hrany vcházející do cílového uzlu nelze zničit, což modelujeme nastavením kapacit a cen za zničení na hodnoty mnohonásobně převyšující možné využití kapacit

i dostupné zdroje stejně, jako v podkapitole 5.1.



Obrázek 5.4: Řešení modelové úlohy pro model 5.2.

Výsledek uvedeme také formou výstupního souboru z programu GAMS, který následně okomentujeme.

```
---------------------------- output EBDM2.gms ----------------------------
---- 130 VARIABLE alpha.L
n7 1.000,     n8 1.000,     n9 1.000,     t 1.000

---- 130 VARIABLE beta.L
           n7          n8          n9
 n4      0.500       0.500
 n5      0.500
 n6                  1.000       1.000

---- 130 VARIABLE gamma.L
                1           2
n4.n7       1.000
n4.n8       1.000
n5.n7                   1.000
n5.n8       1.000       1.000
n5.n9       1.000       1.000

---- 130 VARIABLE iflow.L
           n7          n8          n9
n4       6.500       5.000
n5       3.000      14.000      15.000

---- 130 VARIABLE z2.L     =     25.500
```

Řešení modelu na příslušném obrázku 5.4 vyznačují obarvené hrany, jejichž obarvení určují hodnoty proměnných γ_{ijk}. Žlutě obarvené hrany (n_4, n_7) a (n_4, n_8) představují zničení hrany zcela pomocí zdroje 1, tj. pro tyto hrany platí $\gamma_{ij1} = 1$. Oranžově obarvená hrana (n_5, n_7) je zničena pomocí zdroje 2 a červeně obarvené hrany (n_5, n_8) a (n_5, n_9) jsou zničeny zcela pomocí obou zdrojů zároveň. Konkrétní hodnoty proměnných γ_{ijk} jsou uvedeny ve výpisu z programu GAMS. Vidíme také, že zničení hrany s použitím pouze jednoho zdroje má za následek, že hodnota proměnné β_{ij} je rovna jedné polovině. Ma-

ximální možný tok v síti je napadením redukován z 69 na 25,5. Model i s komentářem a řešení lze nalézt v přiloženém souboru *EBDM2.gms*.

5.3. Model pro přerušování hran s více zdroji, zdrojovými a cílovými uzly

Dalším rozšířením předchozích modelů 5.1 a 5.2 může být případ, kdy se v síti objevuje více počátečních a více cílových uzlů. V takovém případě označíme množinu všech počátečních uzlů S a množinu všech cílových uzlů T. Standardní způsob, jak se v navazující úloze o maximálním toku vypořádat s přítomností více zdrojových a více cílových destinací, je zavedení tzv. super-zdrojového uzlu S' a super-cílového uzlu T'. Super-zdroj S' je uzel spojený se všemi počátečními uzly $s_i \in S$ hranami s nekonečnými kapacitami. Obdobně jsou cílové uzly $t_i \in T$ spojeny hranami s nekonečnými kapacitami se super-zdrojem T'. Zde dostává větší význam jinak zapsané omezení (5.2) z modelu uvedeného v podkapitole 5.1. Navíc se zde předpokládá, že množina počátečních uzlů S a množina cílových uzlů T neobsahuje super-zdrojový uzel S', resp. super-cílový uzel T'. První úroveň dvouúrovňové úlohy se řeší bez zavedení super-zdrojového uzlu S' a super-cílového uzlu T', avšak pro navazující úroveň se jedná o podstatné zjednodušení při modelování úlohy, a proto nelze tuto poznámku vynechat. Navazující úroveň o maximálním toku totiž optimalizuje součet toků na hranách vstupující do super-zdrojového uzlu T'. Formulace modelu s více zdrojovými a více cílovými uzly, která je velice podobná předchozímu modelu z podkapitoly 5.2, je následující

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\alpha_i - \alpha_j + \beta_{ij} + \frac{\gamma_{ijk}}{|K|} \geq 0, \qquad \forall (i,j) \in A, \tag{5.8}$$

$$\alpha_s = 0, \qquad \forall s \in S', \tag{5.9}$$

$$\alpha_t = 1, \qquad \forall t \in T', \tag{5.10}$$

$$\sum_{(i,j)\in A} c_{ijk}\gamma_{ijk} \leq R_k, \qquad \forall k \in K,$$

$$\alpha_i \in \{0,1\}, \qquad \forall i \in N,$$

$$0 \leq \beta_{ij} \leq 1, \qquad \forall (i,j) \in A,$$

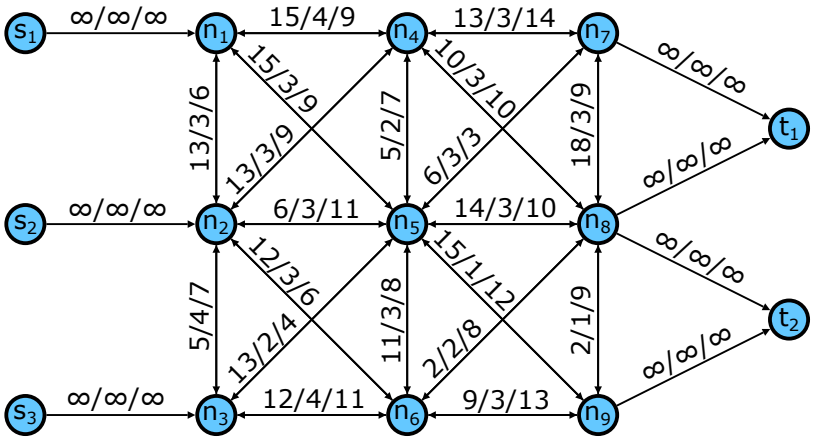
$$0 \leq \gamma_{ijk} \leq 1, \qquad \forall (i,j) \in A, \forall k \in K.$$

Oproti předchozí modelům z podkapitol 5.1 a 5.2 je zde rozdíl v tom, že neuvažujeme minimalizaci řezu mezi dvěma uzly v síti, ale mezi dvěma skupinami uzlů. Tuto skutečnost v GAMSu modelujeme pomocí následujících omezení.

```
stc1(s).. alpha(s) =e= 0;
stc2(t).. alpha(t) =e= 1;
```

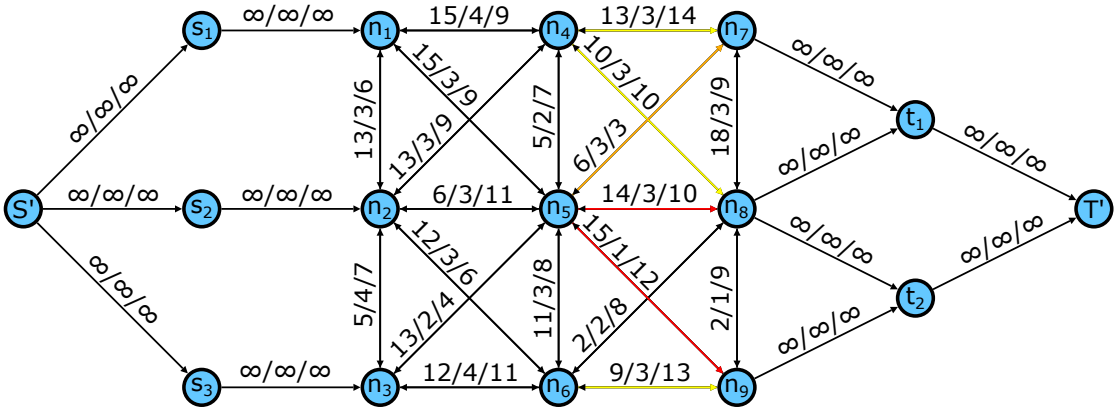
První omezení zastupuje omezení (5.9) z modelu napadení sítě a je platné pro všechny počáteční uzly z množiny S, kterou v GAMSu značíme `s`. Druhé omezení reprezentuje omezení (5.10) a platí pro všechny cílové uzly z množiny T v GAMSu označené jako `t`.

Uvažujme následující modelovou úlohu, která je oproti předchozí rozdílná v tom, že uvažujeme více cílových a počátečních uzlů místo jednoho, ač stále s nekonečnými kapacitami. Čísla na hranách vyjadřují stále totéž, a to postupně kapacitu hran, množství zdrojů prvního druhu a množství zdrojů druhého druhu potřebných pro přerušení příslušných hran.



Obrázek 5.5: Modelová úloha pro model 5.3.

Takto zadanou úlohu lze řešit výše uvedeným modelem, ale jak jsme dříve uvedli, pro navazující úlohu o maximálním toku je třeba síť doplnit o super-zdrojový uzel S', super--cílový uzel T' a příslušné hrany s nekonečnými kapacitami a cenami za přerušení v případě obou potřebných zdrojů. Doplněné zadání úlohy by pak bylo možné graficky vyjádřit následující sítí. V předchozí podkapitole 5.2 jsme uvažovali zadání úlohy s rozpočtem útočníka $R = (10, 25)$. V takovém případě všechny hodnoty proměnné γ_{ijk} nabývaly hodnot 0, nebo 1 pro $\forall k \in K$ a všechny hrany $(i, j) \in A$. Záměrně proto modifikujeme útočníkův rozpočet tak, že $R = (12, 20)$, abychom získali i řešení, kde budou proměnné γ_{ijk} nabývat i jiných hodnot než pouze nula a jedna.



Obrázek 5.6: Řešení modelové úlohy pro model 5.3.

Barvy na obrázku 5.6 opět znázorňují, kterými zdroji byly hrany napadeny. Žluté hrany (n_4, n_7), (n_4, n_8) a (n_6, n_9) jsou napadeny pomocí zdroje 1. Oranžová hrana (n_5, n_7) je hrana napadená pomocí zdroje 2. Červené hrany (n_5, n_8) a (n_5, n_9) jsou napadeny pomocí obou zdrojů. Konkrétní hodnoty nejen pro poničené hrany uvádí následující výpis z programu GAMS.

```
---------------------------- output EBDM3.gms ----------------------------
---- 157 VARIABLE alpha.L
n7 1.000,   n8 1.000,   n9 1.000,   t1 1.000,   t2 1.000,   T 1.000

---- 157 VARIABLE beta.L
            n7          n8          n9
n4       0.500       0.500
n5       0.500                   0.208
n6                   1.000       0.667

---- 157 VARIABLE gamma.L
                1           2
n4.n7       1.000
n4.n8       1.000
n5.n7                   1.000
n5.n8       1.000       1.000
n5.n9       1.000       0.583
n6.n9       0.667

---- 157 VARIABLE iflow.L
            n7          n8          n9
n4       6.500       5.000
n5       3.000      14.000      11.875
n6                              3.000

---- 157 VARIABLE z2.L     =     25.625
```

Konkrétní hodnoty proměnných γ_{ijk} uvádí výše uvedený výpis. Výpočet residuální kapacity hrany po napadení sítě útočníkem vychází logicky z omezení (5.8). Zjednodušeně řečeno, vypočítá se hodnota zlomku $\gamma_{ij} = \sum_k \frac{\gamma_{ijk}}{|K|}$, kterou pak vynásobíme s kapacitou příslušné hrany (i, j) jako $(1 - \gamma_{ij})u_{ij}$. Další možností by bylo prosté vynásobení hodnoty β_{ij} s kapacitou hrany jako $\beta_{ij}u_{ij}$. Maximální tok je zde redukován na 25,625. Numerické výsledky zničení jednotlivých hran a celý výpočet lze nalézt taktéž v přiloženém souboru $EBDM3.gms$.

5.4. Model pro přerušování uzlů

Kromě přerušování hran lze samozřejmě ničit také uzly. Prvotní přístup k přerušování uzlů spočíval v rozdělení uzlu n na pomocné uzly n' a n'' a přiřazení pomocné hrany (n', n''). S užitím této transformace pak přerušení uzlu n představovalo přerušení pomocné hrany (n', n''). V nejhorším možném případě došlo ke zdvojnásobení množství uzlů v modelu a navíc se objevilo až n hran navíc, pokud všech n uzlů mohlo být přerušeno. Účelová funkce modelu zůstává stále stejná. Místo matice cen za přerušení hran $\mathbf{C}$ zavádíme vektor konstantních cen za přerušení uzlů $\mathbf{d} = (d_i)$. I v tomto a následujících případech předpokládáme, že počáteční ani cílový uzel nelze přerušit, tedy cena za jejich přerušení je nekonečná. Klíč k samotnému přerušování uzlů je skryt v omezeních. Model lze zapsat následujícím způsobem:

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\begin{aligned}
\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} &\geq 0, && \forall (i,j) \in A, && (5.11)\\
\alpha_t &= 1, \\
\alpha_s &= 0, \\
\sum_{(i,j)\in A} d_i\delta_i &\leq R, && && (5.12)\\
\gamma_{ij} &= \delta_i, && \forall (i,j) \in A, \forall i \in N, && (5.13)\\
\alpha_i, \delta_i &\in \{0,1\}, && \forall i \in N, \\
\beta_{ij}, \gamma_{ij} &\in \{0,1\}, && \forall (i,j) \in A,
\end{aligned}$$

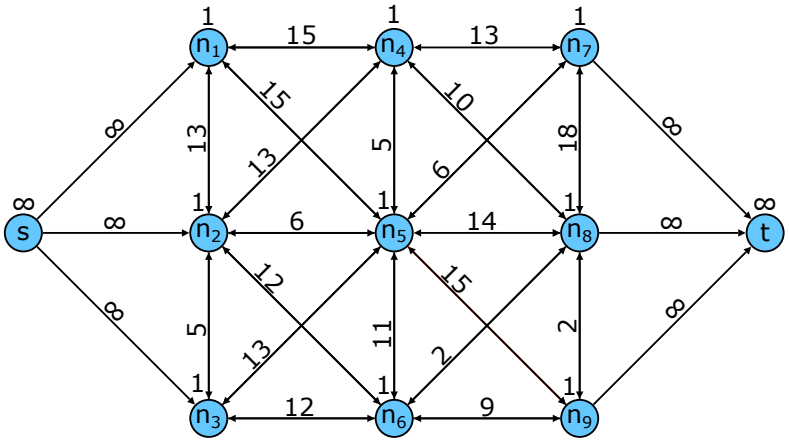
kde

$$\delta_i = \begin{cases} 1 & \text{pokud je uzel } i \text{ zničen,} \\ 0 & \text{jinak.} \end{cases}$$

Tvar a význam omezení (5.11) zůstává stále stejný díky definici klíčových omezení (5.12) a (5.13). První zmíněné omezení říká, kolik uzlů lze přerušit, aniž by útočník překročil svůj rozpočet. Druhé omezení provazuje zničení uzlů s vystupujícími hranami. Je-li uzel zničen, jsou s ním zničeny i z něj vystupující hrany. Díky omezení (5.13) je vytvořena „slepá ulička“, kterou pro přepravu do cílového uzlu nelze využít. Útočník se bude snažit ničit uzly, z nichž vychází hrany náležící do minimálního řezu. Je možné zavést ekvivalentní omezení typu $\gamma_{ij} = \delta_j$, místo omezení (5.13), které by zamezilo vstupu do zničeného uzlu. Uvažování takového omezení navíc by však bylo nadbytečné. Zbývá otázka, jak omezení (5.12) a (5.13) zapsat v programu GAMS. Omezení modelujeme takto:

```
bud.. sum(i, c(i) * delta(i)) =l= budget;
nec(i,j)$(u(i,j)>0).. gamma(i,j) =e= delta(i);
```

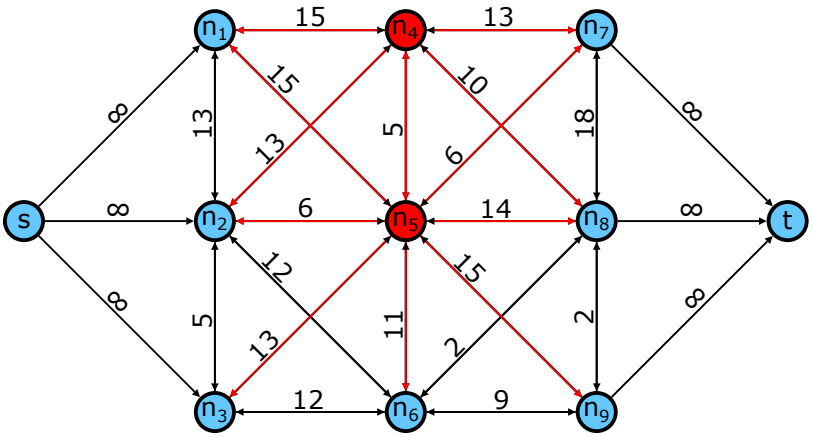
V omezení `bud` odpovídá cena `c(i)` cenně $d(i)$ v omezení (5.12). Omezení `nec(i,j)` v programu GAMS zastupuje omezení (5.13). Zadání úlohy je následující:



Obrázek 5.7: Zadání modelové úlohy pro model 5.4.

Uvažujeme stejnou modelovou síť jako v podkapitole 5.1. Předpokládejme, že cena za zničení všech vnitřních uzlů je jedna. Cena zničení uzlů s a t je nastavena na číslo mnohonásobně vyšší, než je rozpočet útočníka, aby je útočník nemohl zničit, v GAMSu 1000. Výsledky získané z programu *Nodal_deletion.gms* jsou zapsány níže, následuje grafická interpretace řešení a komentář.

```
---------------------- output Nodal_deletion.gms ----------------------
---- 106 VARIABLE alpha.L
n7 1.000,     n8 1.000,     n9 1.000,     t 1.000

---- 106 VARIABLE beta.L
            n8          n9
n6       1.000       1.000


---- 106 VARIABLE delta.L
n4 1.000,     n5 1.000

---- 106 VARIABLE gamma.L
           n1          n2          n3          n4          n5          n6
n4      1.000       1.000                                1.000
n5      1.000       1.000       1.000       1.000                   1.000

 +         n7          n8          n9
n4      1.000       1.000
n5      1.000       1.000       1.000

   ---- 106 VARIABLE iflow.L
           n1          n2          n3          n4          n5          n6
n4      15.000      13.000                                5.000
n5      15.000       6.000      13.000       5.000                  11.000

 +          n7          n8          n9
n4      13.000      10.000
n5       6.000      14.000      15.000

---- 106 VARIABLE z2.L     =     11.000
```



Obrázek 5.8: Řešení modelové úlohy pro model 5.4.

Podle hodnoty proměnných δ_i je řešením modelové úlohy zničení uzlů n_4 a n_5. Zničení těchto uzlů potvrzuje, co bylo napsáno dříve. Útočník ničí uzly, z nichž vychází hrany náležící do minimálního řezu. Protože jsme uvažovali síť z podkapitoly 5.1, můžeme porovnat výsledky. Zde, stejně jako ve zmíněné podkapitole, je minimální řez reprezentován množinou hran vystupující ze skupiny uzlů $\{n_4, n_5, n_6\}$ a vstupující do skupiny uzlů $\{n_7, n_8, n_9\}$. Přerušeny byly v důsledku zničení uzlů i hrany, které do minimálního řezu nenáleží. To však nemá z hlediska navazující úlohy o maximálním toku žádný vliv. V podkapitole 5.1 útočník napadením sítě zničil hrany (n_4, n_7), (n_5, n_8) a (n_5, n_9). Zničení uzlů n_4 a n_5 je tedy logickým řešením naší úlohy. Jelikož zmíněné uzly nemohou nic propustit, jediný tok v síti zajišťuje uzel n_6. Tok je tak minimalizován z původních 69 na 11. Model i s daty lze nalézt v přiloženém souboru *Nodal_deletion.gms*.

5.5. Model pro přerušování uzlů a hran se sdílenými zdroji

Další rozšíření všech předchozích modelů poskytuje následující model, který umožňuje přerušování hran i uzlů zároveň. Při výpočtu používáme námi definovanou konstantní matici $\mathbf{C} = (c_{ij})$ představující ceny za přerušení hran (i, j) a zároveň konstantní vektor cen za přerušení uzlů $\mathbf{d} = (d_i)$. Předpokládáme, že pro přerušení hrany i uzlu můžeme použít stejný zdroj. Model lze zapsat jako

$$\min \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

$$\begin{aligned}
\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} &\geq 0, && \forall (i,j) \in A, \\
\alpha_t &= 1, \\
\alpha_s &= 0, \\
\sum_{(i,j)\in A} c_{ij}\gamma_{ij} - \sum_{(i,j)\in A} c_{ij}\delta_i + \sum_{i\in N} \delta_i d_i &\leq R, && (5.14) \\
\gamma_{ij} &= \delta_i, && \forall (i,j) \in A, \forall i \in N, \quad (5.15) \\
\alpha_i, \delta_i &\in \{0,1\}, && \forall i \in N, \\
\beta_{ij}, \gamma_{ij} &\in \{0,1\}, && \forall (i,j) \in A.
\end{aligned}$$

Omezení (5.15), díky kterému jsou při zničení uzlů přerušeny i hrany, které z uzlu vychází, stále platí. Zásadní změnou oproti předchozímu modelu je omezení (5.14) nahrazující omezení (5.12) z předchozího modelu. První suma reprezentuje cenu všech přerušených hran, nezávisle na tom, zda byly přerušeny v důsledku zničení uzlu či jinak. Druhá suma reprezentuje cenu přerušených hran, které vychází ze zničených uzlů. Třetí suma reprezentuje cenu zničených uzlů. Zásadní roli hraje v pořadí druhá suma v omezení (5.14), protože snižuje cenu útoku na síť o cenu „automaticky“ přerušených hran v důsledku přerušení příslušných uzlů. Jestliže je tedy zničen nějaký konkrétní uzel n_i, cena zničených hran, které z něho vychází, není započítávána do ceny útoku, respektive je z něj odečítána. Nemůže tak nastat situace, kdy by byla cena zničené hrany započítána

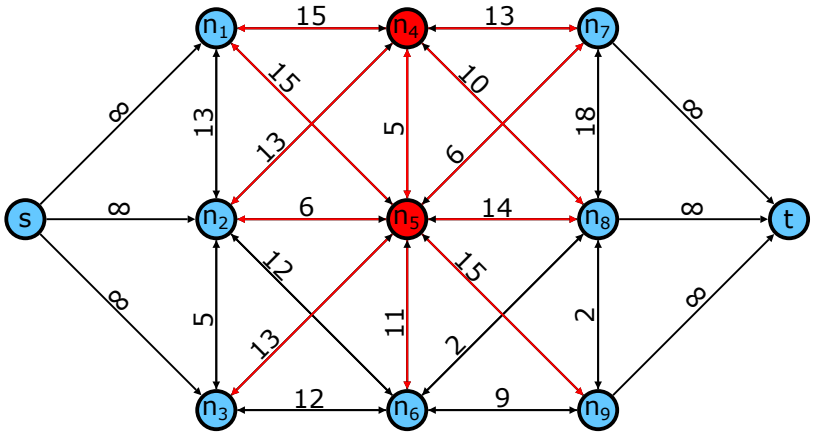
dvakrát. V GAMSu zapíšeme omezení (5.14) a (5.15) následujícím způsobem:

```
bud.. sum((i,j),c2(i,j)*gamma(i,j)) - sum((i,j),c2(i,j)*delta(i)) +
sum(i,delta(i)*c1(i)) =l= budget;
nec(i,j)$(u(i,j)>0).. gamma(i,j) =g= delta(i);
```

Uvažujme stejnou síť jako v podkapitolách 5.1 a 5.4. Navíc obdobně jako v předchozí podkapitole předpokládejme, že cena za přerušení všech uzlů mimo počátečního s a koncového t je rovna jedné. Opět bude útočníkův rozpočet $R = 2$. Následující výsledky poskytuje program *Nodal_deletion_shared.gms*.

```
--------------------- output Nodal_deletion_shared.gms --------------------
---- 122 VARIABLE alpha.L
n7 1.000,     n8 1.000,     n9 1.000,     t 1.000

---- 122 VARIABLE beta.L
            n8          n9
n6       1.000       1.000


---- 122 VARIABLE delta.L
n4 1.000,     n5 1.000

---- 122 VARIABLE gamma.L
           n1          n2          n3          n4          n5          n6
n4      1.000       1.000                                1.000
n5      1.000       1.000       1.000       1.000                   1.000

 +         n7          n8          n9
n4      1.000       1.000
n5      1.000       1.000       1.000

---- 122 VARIABLE iflow.L
            n1          n2          n3          n4          n5          n6
n4      15.000      13.000                                5.000
n5      15.000       6.000      13.000       5.000                   11.000

 +          n7          n8          n9
n4      13.000      10.000
n5       6.000      14.000      15.000

---- 122 VARIABLE z2.L     =     11.000
```

Výsledky lze taktéž prezentovat graficky pomocí následujícího obrázku.



Obrázek 5.9: Řešení modelové úlohy pro model 5.5.

Vzhledem k charakteru úlohy je zde podle hodnot proměnných δ_{ij} optimálním řešením přerušení uzlů n_4 a n_5 a s nimi přerušení z nich vystupujících hran. Cena za přerušení hran i uzlů je v obojím případě rovna jedné, a tedy je logičtější ničit uzly, neboť zničením uzlu je zničeno více hran zároveň. Jelikož jsou hrany mezi vnitřními uzly sítě uvažovány jako obousměrné, přerušením vnitřních uzlů dochází i k přerušení některých hran, které do minimálního řezu nepatří. To však nemá žádný vliv na výsledek úlohy. Pro úplně přerušení toku v síti by musel být útočníkův rozpočet R roven třem, aby mohl přerušit i zbývající hrany minimálního řezu vystupující z uzlu n_6. Maximální tok v síti je tedy stejný jako v předchozím případě.

5.6. Model pro přerušování uzlů a hran s nesdílenými zdroji

Nyní se přímo nabízí otázka, jak modelovat situaci, v níž jsou pro přerušování hran a uzlů potřeba různé zdroje, například lodě na moři a auta na silnicích. Útočník potřebuje dva různé, na sobě nezávislé rozpočty. Podobnou situaci lze zapsat následujícím modelem.

$$\text{Min} \sum_{(i,j)\in A} u_{ij}\beta_{ij},$$

za podmínek

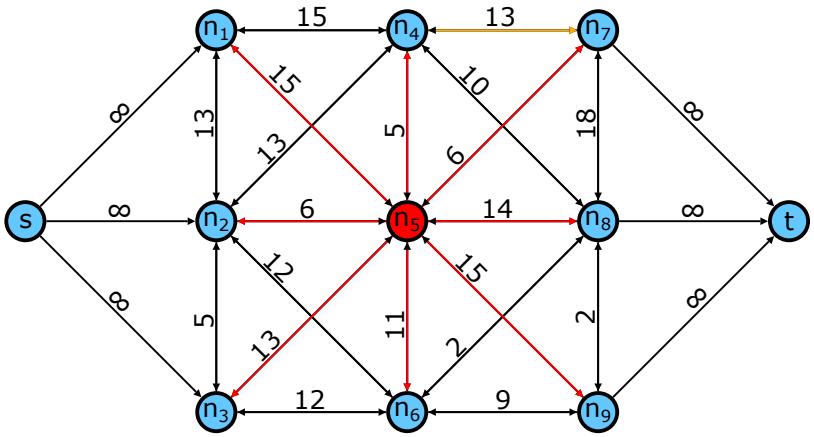
$$\begin{aligned}
\alpha_i - \alpha_j + \beta_{ij} + \gamma_{ij} &\geq 0, && \forall (i,j) \in A, \\
\alpha_t &= 1, \\
\alpha_s &= 0, \\
\sum_{(i,j)\in A} d_i\delta_i &\leq R_1, && (5.16) \\
\sum_{(i,j)\in A} c_{ij}\gamma_{ij} - \sum_{(i,j)\in A} c_{ij}\delta_i &\leq R_2, && (5.17) \\
\gamma_{ij} &= \delta_i, && \forall (i,j) \in A, \forall i \in N, \\
\alpha_i, \delta_i &\in \{0,1\}, && \forall i \in N, \\
\beta_{ij}, \gamma_{ij} &\in \{0,1\}, && \forall (i,j) \in A.
\end{aligned}$$

Omezení (5.16) má stejný význam jako v modelu z podkapitoly 5.5, tedy cena přerušených uzlů nepřesáhne útočníkův rozpočet R_1. Omezení (5.17) je pak velice podobné omezení (5.14). Opět od ceny přerušených hran odečítáme cenu hran přerušených v důsledku přerušení uzlu. Při podrobném pohledu vidíme, že sečtením omezení (5.16) a omezení (5.17) bychom dostali omezení (5.14) z předchozí podkapitoly 5.5. Omezení (5.16) a (5.17) zapíšeme v GAMSu následujícím způsobem:

```
bud1.. sum(i, c1(i) * delta(i)) =l= budget1;
bud2.. sum((i,j),c2(i,j)*gamma(i,j)) - sum((i,j),c2(i,j)*delta(i)) =l=
budget2;
```

Uvažujme síť jako v předchozí podkapitole. Obdobně i zde cena za přerušení všech hran mimo těch, které začínají v počátečním, resp. končí v cílovém uzlu, je rovna jedné. Stejně tak zničení každého uzlu kromě počátečního a cílového má cenu jedna. Předpokládejme navíc, že $R_1 = 1$ a $R_2 = 1$. Nejprve uvedeme výsledky získané z programu *Nodal_deletion_different.gms*, které pak uvedeme názorně graficky na obrázku 5.10 i s komentářem.

```
------------------ output Nodal_deletion_different.gms ------------------
---- 124 VARIABLE alpha.L
n7 1.000,    n8 1.000,     n9 1.000,     t 1.000

---- 124 VARIABLE beta.L
            n8          n9
n4       1.000
n6       1.000       1.000

---- 124 VARIABLE delta.L
n5 1.000

---- 124 VARIABLE gamma.L
           n1          n2          n3          n4          n6          n7
n4                                                                   1.000
n5      1.000       1.000       1.000       1.000       1.000       1.000

+          n8          n9
n5      1.000       1.000

---- 124 VARIABLE iflow.L
            n1          n2          n3          n4          n6          n7
n4                                                                  13.000
n5      15.000       6.000      13.000       5.000      11.000       6.000

 +          n8          n9
n5      14.000      15.000

---- 124 VARIABLE z2.L     =   21.000
```



Obrázek 5.10: Řešení modelové úlohy pro model 5.6.

Řešením takto zadané úlohy je podle hodnoty proměnné δ_5 zničení uzlu n_5 a z něj vycházejících červeně označených hran. V důsledku přerušení uzlu n_5 se přerušují i hrany, které nenáleží do minimálního řezu. Celková kapacita hran, které patří do minimálního řezu a jsou zničeny v důsledku zničení uzlu n_5 je 35. Dále se útočník rozhodne pro přerušení oranžově označené hrany (n_4, n_7) s kapacitou 13. Obdobně jako v předchozích dvou podkapitolách a podkapitole 5.1, útočník cílí na hrany vycházející ze skupiny uzlů $\{n_4.n_5, n_6\}$ a vstupující do skupiny uzlů $\{n_7, n_8, n_9\}$. Útok minimalizoval tok v síti na 21. Pokud by chtěl útočník zamezit toku v síti úplně, musel by navýšit rozpočet R_1 o 2 nebo rozpočet R_2 o 3.

6. Stochastické napadení sítě

V této kapitole se budeme věnovat napadení sítě s využitím deterministických reformulací uvedených v podkapitole 4.1. V předchozí kapitole 5 jsme uvažovali defenzivní přístup, kdy jsme očekávali nejhorší možný scénář, který spočíval v tom, že cílem napadajícího je minimalizovat tok v síti bez ohledu na jakékoli další motivy. Co ale dělat v případě, že další potenciální motivy napadajícího nejsou známé? Co když se napadající chová náhodně? Co když vnímáme pouze náhodné projevy nějaké neracionální bytosti?

6.1. WS přístup k napadení sítě

Uvažujme nejprve přístup wait-and-see, který si můžeme představit tak, že vyšleme kolegy, aby zjistili stav sítě a rozmístění silničních kontrol. Se získanou znalostí o stavu sítě pak můžeme neodhaleni převážet kontraband. Můžeme sice převážet méně, než kdyby síť nebyla kontrolována, ale pro nás je důležité zůstat v utajení a nepřijít o zboží. Obdobně lze uvažovat v případě přírodních katastrof, kdy při silných deštích či zemětřesení jsou náhodně podemlety některé horské stezky.

Připomeňme, jak vypadá deterministická reformulace wait-and-see z podkapitoly 4.1, kterou upravíme pro maximalizační úlohu:

$$\begin{aligned} \max \quad & f\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big), \\ \text{za podmínek} \quad & g\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big) \leq \mathbf{0}, \\ & h\big(\mathbf{x}(\boldsymbol{\xi}), \boldsymbol{\xi}\big) = \mathbf{0}, \\ & \mathbf{x}(\boldsymbol{\xi}) \in X. \end{aligned}$$

Jak jsme zmínili, naším cílem bude maximalizace toku přes napadenou síť, jejíž napadení se nám může jevit jako náhodné, protože neznáme další případné motivy napadajícího. Maximalizaci toku v síti budeme opět modelovat pomocí maximalizace toku přes návratovou hranou s dostatečně velkou kapacitou, viz podkapitola 3.2. Využijeme stejný model jako ve zmíněné podkapitole, ale s drobnou úpravou, protože budeme předpokládat několik scénářů, které mohou nastat. Zde budeme opět uvažovat množinu uzlů $N' = N \setminus \{s, t\}$. Uvažujeme množinu realizací napadení sítě $\mathcal{S} = \{1, ..., S\}$. Využijeme upravený model, který řešíme pro každou realizaci $r \in \mathcal{S}$.

$$\max x^r_{ts},$$

za podmínek

$$\sum_{j \in N'} x^r_{sj} - \sum_{i \in N'} x^r_{is} - x^r_{ts} \leq 0, \tag{6.1}$$

$$\sum_{j \in N} x^r_{nj} - \sum_{i \in N} x^r_{in} \leq 0, \qquad \forall n \in N', \tag{6.2}$$

$$\sum_{j \in N'} x^r_{tj} - \sum_{i \in N'} x^r_{it} + x^r_{ts} \leq 0, \tag{6.3}$$

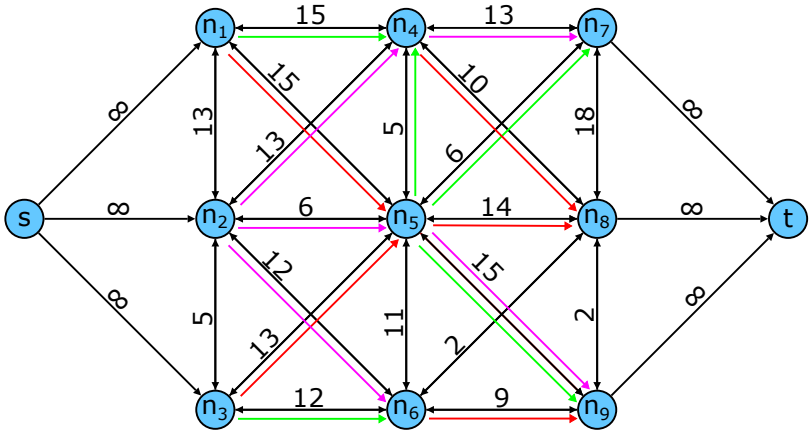
$$0 \leq x^r_{ij} \leq u^r_{ij}, \qquad \forall (i,j) \in A. \tag{6.4}$$

Náhoda do úlohy vstupuje tak, že mění horní meze u_{ij}^r pro tok na hraně. V případě, že je hrana napadena, její kapacita klesne na nulu a nelze přes ní nic převézt. Ukážeme, jak modelujeme výpočet pro deterministickou WS reformulaci a tři různé scénáře v programu GAMS pomocí cyklického výpočtu.

```
model WS / obj, bal, src, ter, xc /;
loop(ss, gamma(i,j) = gammas(i,j,ss);
solve WS max z using mip; xs(i,j,ss) = x.l(i,j); zws(ss) = z.l);
display gammas, xs, zws;
```

Model je pojmenován `WS` a skládá se z dříve užitých tokových omezení (6.1) - (6.4), která jsou v programu GAMS modelována stejně, jak jsme uvedli v podkapitole 5.1. Protože uvažujeme pouze jednu úroveň problému, vynecháváme v deklaracích proměnných a omezení oproti podkapitole 5.1 v jejich názvech číslo „2“. Různé realizace náhodného napadení jsou modelovány pomocí příkazu `loop`, který opakuje výpočet pro všechny scénáře r, které v GAMSu označujeme `ss`. Parametry `xs(i,j,ss)` a `zws(ss)` slouží k zápisu řešení modelu pro jednotlivé scénáře. Pomocí příkazu `display` pak GAMS zobrazí v .lst souboru výsledky.

Uvažujme modelovou úlohu z podkapitoly 5.1, kde čísla na hranách představují jejich kapacity a kde je maximální tok v nenapadené síti 69. Dále uvažujme tři scénáře, které mohou nastat. Tři uvažované scénáře odlišíme barvami.



Obrázek 6.1: Modelová úloha z podkapitoly 5.1 s vyznačenými scénáři napadení.

První realizace je vyznačena zelenými šipkami. Hrany, které jsou pro tento scénář napadeny a zneprůchodněny jsou (n_1, n_4), (n_3, n_6), (n_5, n_4), (n_5, n_7), (n_5, n_9). Druhá realizace je vyznačena červenými šipkami. Pro tuto realizaci jsou napadeny hrany (n_1, n_5), (n_3, n_5), (n_4, n_8), (n_5, n_8), (n_6, n_9). Třetí realizace je vyznačena fialovými šipkami a pro tuto realizace jsou přerušeny hrany (n_2, n_4), (n_2, n_5), (n_2, n_6), (n_4, n_7), (n_5, n_9). Nyní uvedeme výsledky získané z programu *WS.gms*. Každý sloupec odpovídá jednomu scénáři. Parametr `xs` vyjadřuje tok na hranách, které jsou uvedeny v levém sloupci, tedy například `s .n1` představuje tok na hraně (s, n_1). Hodnoty parametru `zws` pak představují hodnotu účelové funkce pro jednotlivé scénáře.

```
----    117 PARAMETER xs

                1           2           3
s .n1       19.000      18.000      21.000
s .n2        1.000       1.000
s .n3       18.000      17.000      20.000
n1.n2       13.000      13.000       1.000
n1.n4                    5.000       5.000
n1.n5        6.000                  15.000
n2.n3                                5.000
n2.n4       13.000      13.000
n2.n5                    5.000
n2.n6       12.000       1.000
n3.n2        5.000       5.000
n3.n5       13.000                  13.000
n3.n6                   12.000      12.000
n4.n5                    5.000
n4.n7       13.000      13.000
n4.n8                                10.000
n5.n2        6.000                   4.000
n5.n4                                5.000
n5.n7                    6.000       6.000
n5.n8       14.000                  14.000
n5.n9                   15.000
n6.n5        1.000      11.000       1.000
n6.n8        2.000       2.000       2.000
n6.n9        9.000                   9.000
n7.n8       13.000
n7.t                    23.000      24.000
n8.n7                    4.000      18.000
n8.n9                                2.000
n8.t        31.000                   6.000
n9.n8        2.000       2.000
n9.t         7.000      13.000      11.000
t .s        38.000      36.000      41.000

----    117 PARAMETER zws
1 38.000,    2 36.000,    3 41.000
```

Hodnota účelové funkce pro zelenou realizaci je 38, pro červenou realizaci 36 a pro fialovou realizaci 41. Model včetně dat lze nalézt v programu *WS.gms* v příloze. Zároveň byl model testován i na větších sítích a pro více scénářů, aby byla ověřena jeho funkčnost. Větší sítě pro 10 scénářů a 30 vnitřních uzlů a pro 30 scénářů a 100 vnitřních uzlů lze najít v přiložených programech *WS10x30.gms* a *WS30x100.gms* a po jejich spuštění jsou získány výsledky ve formě .lst souboru.

Otázkou zůstává „co dělat“, když nemáme kolegy, které bychom mohli vyslat zjistit stav sítě a rozmístění kontrol. V takovém případě je třeba aplikovat here-and-now přístup.

6.2. HN přístup k napadení sítě

V této podkapitole budeme využívat deterministické reformulace, které jsme uvedli v podkapitole 4.1. Důvodů pro uvažování here-and-know přístupu může být hned několik. Uvažujme případ, kdy se policejní kontroly pohybují v síti od města k městu a prohlíží továrny. K naší továrně, kde vyrábíme převážený kontraband, se nezadržitelně blíží a my musíme rychle rozhodnout, kterou cestou se pokusíme kontraband k zákazníkovi převézt. Další možností je, že náš zákazník potřebuje akutně extra dodávku zboží a my musíme rychle rozhodnout, kudy se vydat na cestu, přestože víme, že v síti probíhají kontroly, o jejichž rozmístění nemáme buď žádnou informaci, nebo máme informaci jen o dřívějších rozmístěních.

6.2.1. IS přístup k napadení sítě

Představme si situaci, kdy je potřeba akutně přepravit množství kontrabandu z naší výrobny k zákazníkovi. Dále uvažujme stejné tři scénáře jako v předchozí úloze. Naší snahou je vyhnout se policejním kontrolám a převézt maximum zboží. Nezávisle na našem přání však probíhají v síti na hranách kontroly, jejichž cílem je odhalit přepravované nezákonné zboží. Musíme se rozhodnout, přes které hrany zboží přepravit.

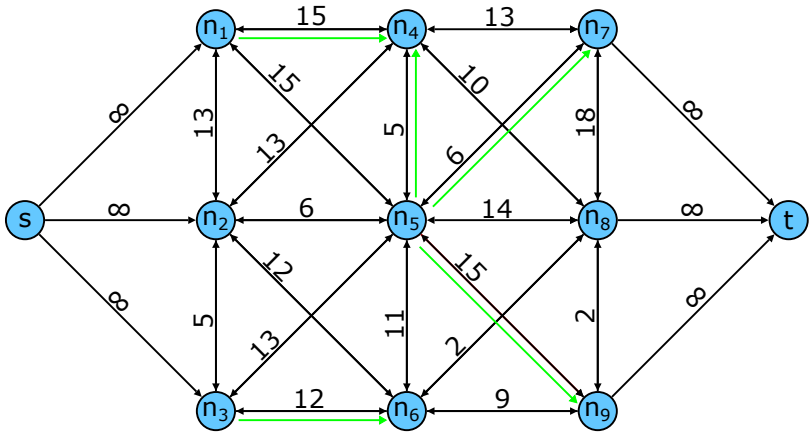
Prvním přístupem k modelování problému v pořadí, v jakém jsme přístupy uvedli v podkapitole 4.1, je uvažování deterministické reformulace IS. Vybereme jednu realizaci $r_0 \in \mathcal{S}$, která se nám může jevit jako nejvhodnější. Budeme předpokládat, že pro tuto realizaci jsou kontroly na hranách v síti rozmístěny podle prvního (zeleného) scénáře. Vozy s kontrabandem tedy vyšleme přes síť tak, aby se těmto pro nás kritickým hranám vyhnuly. Vozy jsou, bohužel, propojeny vysílačkami a GPS lokátory, takže pokud policejní kontrola odhalí jedno vozidlo, odhalí všechna. Pokud budou tedy kontroly rozmístěny jinak, než předpokládáme, a vozidla poputují přes hrany kontrolované policií, budou všechna odhalena a příslušný tok bude nepřípustný. Reformulaci IS pro model maximálního toku zapíšeme takto

$$\max x_{ts},$$

za podmínek

$$\begin{aligned}
\sum_{j\in N'} x_{sj} - \sum_{i\in N'} x_{is} - x_{ts} &\leq 0, \\
\sum_{j\in N} x_{nj} - \sum_{i\in N} x_{in} &\leq 0, \qquad \forall n \in N', \\
\sum_{j\in N'} x_{tj} - \sum_{i\in N'} x_{it} + x_{ts} &\leq 0, \\
0 \leq x_{ij} &\leq u_{ij}^{r_0}, \qquad \forall (i,j) \in A.
\end{aligned}$$

Uvažujme jako nejpravděpodobnější první scénář, tedy $r_0 = 1$. V následujícím obrázku 6.2 je scénář zvýrazněn zelenými souběžnými hranami. Pokud se u hrany vyskytuje zelená souběžná hrana, její kapacita $u_{ij}^{r_0}$ je nulová, pro ostatní hrany zůstává kapacita nezměněna, viz čísla nad hranami.

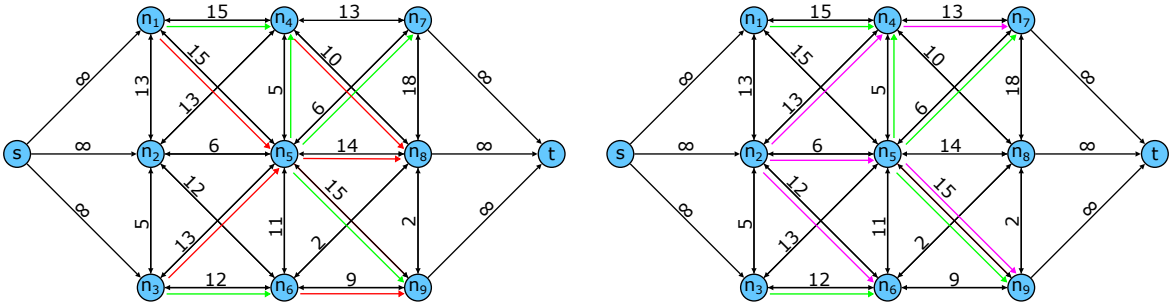


Obrázek 6.2: Modelová úloha z podkapitoly 5.1 se zvýrazněným scénářem 1.

Optimální řešení úlohy pro první scénář $\mathbf{x}^{\mathrm{IS}}_{\max}$ je uvedeno v následující tabulce `x.L` formou výstupního .lst souboru z programu *IS.gms*. Následuje hodnota účelové funkce $z^{\mathrm{IS}}_{\max}$ zapsaná formou parametru `z.L`.

```
----     73 VARIABLE x.L

              s          n1          n2          n3          n4          n5

s                    19.000       1.000      18.000
n1                                13.000                                6.000
n2                                                       13.000
n3                                 5.000                               13.000
n5                                 6.000
n6                                                                      1.000
t         38.000

 +           n6          n7          n8          n9           t

n2        12.000
n4                    13.000
n5                                14.000
n6                                 2.000       9.000
n7                                13.000
n8                                                       31.000
n9                                 2.000                   7.000

----     73 VARIABLE z.L                  =       38.000
```

V následujících grafech na obrázcích 6.3a a 6.3b je zelenou barvou vždy zvýrazněn první scénář a jinou barvou druhý a třetí scénář. Vozidla jedou přes síť podle prvního scénáře tak, jak jsme uvedli ve výsledcích předchozí podkapitoly 6.1. Druhá barva v obrázcích vždy ukazuje reálné rozmístění kontrol v síti.



(a) Kombinace scénářů 1 a 2 (b) Kombinace scénářů 1 a 3

Obrázek 6.3: Modelová úloha uvažovaná pro scénář 1.

Jak se lze přesvědčit při porovnání toku v síti při prvním scénáři a rozmístění kontrol při dalších dvou scénářích, vždy bude alespoň jedno vozidlo při přepravě odhaleno. Tím pádem jsou odhalena všechna auta a pro první scénář jsou řešení obou dvou dalších scénářů nepřípustná. Řešení včetně modelu lze nalézt v přiloženém souboru *IS.gms*, který též umožňuje verifikaci přípustnosti řešení pro jiné scénáře. Zároveň byl model testován i na větších sítích a pro více scénářů, aby byla ověřena jeho funkčnost. Větší sítě pro 10 scénářů a 30 vnitřních uzlů a pro 30 scénářů a 100 vnitřních uzlů lze najít v přiložených programech *IS10x30.gms* a *IS30x100.gms*.

6.2.2. EV přístup k napadení sítě

Uvažujme situaci, že jsme v polovině kalendářního roku a policejní kontroly v modelové síti začal od počátku roku koordinovat nový aktivní policejní šéf. Od začátku roku proběhly v síti tři policejní kontrolní akce, jejichž záměrem bylo odhalit dodávky zakázaného zboží dováženého skrz síť. Každou ze tří policejních akcí můžeme považovat považovat za scénář či realizaci r náhodného vektoru $\boldsymbol{\xi}$.

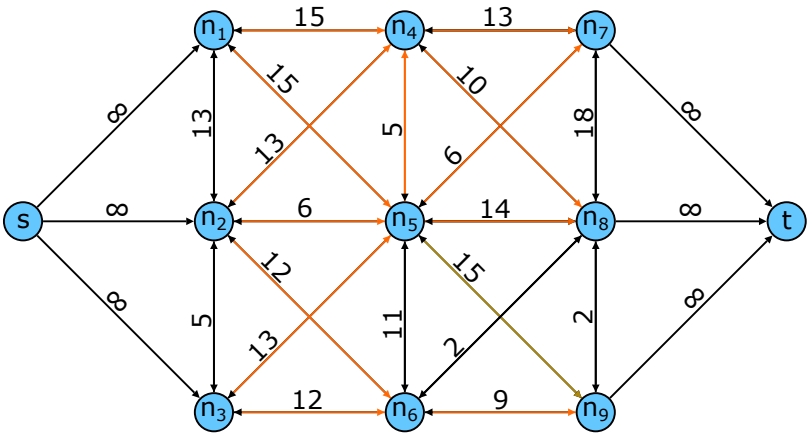
Víme, že v současné době v síti probíhá opět policejní akce, kontroly jsou náhodně rozmístěny a my jejich konkrétní pozice neznáme a nemáme ani způsob, jak je zjistit. Uvažujme stejné tři scénáře jako v podkapitole 6.1, kde každý scénář reprezentuje jednu z proběhlých policejních kontrolních akcí. Znepokojeni prudkým nárůstem policejních kontrol a akutní potřebou expedice vyprodukovaného kontrabandu jsme donuceni učinit okamžité rozhodnutí. Zprůměrujme objemy policejních kontrol, které vytvoří nové horní meze pro tok na hranách. Model zapíšeme následujícím způsobem

$$\max x_{ts},$$

za podmínek

$$\begin{aligned}
\sum_{j\in N'} x_{sj} - \sum_{i\in N'} x_{is} - x_{ts} &\leq 0, \\
\sum_{j\in N} x_{nj} - \sum_{i\in N} x_{in} &\leq 0, \qquad \forall n \in N', \\
\sum_{j\in N'} x_{tj} - \sum_{i\in N'} x_{it} + x_{ts} &\leq 0, \\
0 \leq x_{ij} &\leq \sum_{r\in\mathcal{S}} p_r u_{ij}^r, \qquad \forall (i,j) \in A,
\end{aligned}$$

kde $p_r = \frac{1}{3}, \forall r \in \mathcal{S}$. Naše rozhodnutí zachycuje následující obrázek 6.4.



Obrázek 6.4: Modelová úloha z podkapitoly 5.1 s vyznačenými průměrnými kontrolami.

V obrázku jsou barevně odlišeny tři typy hran. Nad hranami je vždy uvedena původní neredukovaná kapacita. Prvním typem jsou černé hrany, které nebyly kontrolovány a u nichž předpokládáme, že kontrolovány v současné době nejsou a jejich kapacita se tedy zachovává. Světle hnědé hrany byly kontrolovány právě jednou z předchozích tří policejních kontrolních akcí. Okrová hrana (n_5, n_9) byla kontrolována během dvou policejních akcí. Máme tři realizace, jimž přiřazujeme stejnou pravděpodobnost rovnu jedné třetině. V rámci těchto úvah platí, jestliže byla hrana kontrolována právě jednou, pak její kapacita klesla o třetinu, pokud dvakrát, pak klesla o dvě třetiny. Pokud předpokládáme, že se takto redukují kapacity hran v síti, chceme pak maximalizovat množství přepraveného kontrabandu. I zde, stejně jako v podkapitole 5.1, platí, že maximální tok v síti je roven 69. Získané výsledky opět uvedeme formou části .lst souboru získaného v GAMSu pomocí programu *EV.gms*.

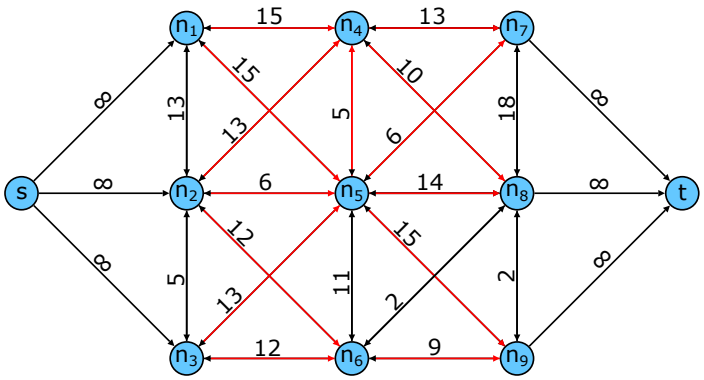
```
----     115 VARIABLE x.L

             s          n1          n2          n3          n4          n5
s                   33.000       6.000       2.667
n1                               13.000                  10.000      10.000
n2                                           5.000       2.000       4.000
n5                                           0.333       3.333
n6                                                                   8.000
t       41.667

 +          n6          n7          n8          n9           t
n2       8.000
n3       8.000
n4                   8.667       6.667
n5                   4.000       9.333       5.000
n6                               2.000       6.000
n7                              12.667
n8                                                       32.667
n9                               2.000                   9.000
```

```
----     115 PARAMETER zev                  =       41.667
```

Hodnoty `x.L` nám říkají, kudy a v jakém množství je převáženo zboží v modelové síti, a odpovídají hodnotám $\mathbf{x}_{\max}^{\mathrm{EV}}$. Proměnná `zev` pak uvádí hodnotu účelové funkce pro transport v síti s redukovanými kapacitami hran a odpovídá $z_{\max}^{\mathrm{EV}}$. Zároveň byl model testován i na větších sítích a pro více scénářů, aby byla ověřena jeho funkčnost. Větší sítě pro 10 scénářů a 30 vnitřních uzlů a pro 30 scénářů a 100 vnitřních uzlů lze najít v přiložených programech *EV10x30.gms* a *EV30x100.gms*.

6.2.3. MM přístup k napadení sítě

Uvažujme znovu předchozí tři scénáře z podkapitoly 6.1. Představme si situaci, kdy se natolik obáváme kontrol, že pokud v některém ze scénářů byla uskutečněna kontrola na nějaké hraně, rozhodneme se tuto hranu nevyužít. Aplikujeme maximálně defenzivní přístup typu „nebezpečí číhá všude“. Připomeňme si modelovou síť, kde jsou červeně obarveny ty hrany, které byly alespoň v jednom ze scénářů napadeny policejní kontrolou.



Obrázek 6.5: Modelová úloha z podkapitoly 5.1.

Uvažujeme vlastně platnost všech tří scénářů najednou. Takový přístup lze modelovat následujícím způsobem

$$\max z,$$

za podmínek

$$\begin{aligned}
z &\le x_{ts}^r, && \forall r \in \mathcal{S}, && (6.5)\\
\sum_{j\in N'} x_{sj}^r - \sum_{i\in N'} x_{is}^r - x_{ts}^r &\le 0, && \forall r \in \mathcal{S}\\
\sum_{j\in N} x_{nj}^r - \sum_{i\in N} x_{in}^r &\le 0, && \forall n \in N', \forall r \in \mathcal{S}\\
\sum_{j\in N'} x_{tj}^r - \sum_{i\in N'} x_{it}^r + x_{ts}^r &\le 0, && \forall r \in \mathcal{S}\\
0 \le x_{ij}^r &\le u_{ij}^r, && \forall (i,j) \in A, \forall r \in \mathcal{S},\\
x_{ij}^1 &= x_{ij}^r, && \forall r \in \mathcal{S}. && (6.6)
\end{aligned}$$

Všechna omezení kromě (6.5) a (6.6) jsou stejná jako v předchozích modelech až na výjimku v tom, že chceme, aby platila pro všechny scénáře současně. Omezení (6.5) propojuje účelové hodnoty pro všechny scénáře. Omezení (6.6) je neanticipativní omezení, které nám říká, že vždy se rozhodneme stejně bez ohledu na volbu scénáře. Pro modelovou úlohu z obrázku 6.5 uvádíme následující řešení formou .lst souboru získaného díky programu *MM.gms*.

```
----     66 VARIABLE xs.L

                    ( ALL       0.000 )

----     66 VARIABLE z.L                 =        0.000
```

V uvedeném řešení z programu GAMS odpovídá `xs.L` hodnotám x_{ij}^r. Scénáře jsou takové, že řez mezi skupinami hran $\{n_1, n_2, n_3\}$ a $\{n_4, n_5, n_6\}$ je přerušen zcela, a tedy přes síť nelze nic přepravit. Hodnota účelové funkce z odpovídá proměnné `z.L` a je nulová. Situace, kdy nemůžeme nic přepravit, je pro nás značně nevýhodná. Co kdyby bylo možné některé policejní jednotky uplatit, aby nás nechali přes kontrolované úseky zboží převézt? Pak by bylo možné přes daný úsek opět zboží převážet, jinými slovy, dovolili bychom zapnutí některých hran. Na tuto úvahu navážeme v následující podkapitole 6.3. Zároveň byl model testován i na větších sítích a pro více scénářů, aby byla ověřena jeho funkčnost. Větší sítě pro 10 scénářů a 30 vnitřních uzlů a pro 30 scénářů a 100 vnitřních uzlů lze najít v přiložených programech *MM10x30.gms* a *MM30x100.gms*.

6.3. Dvoustupňové napadení sítě s kompenzací

Opusťme na chvilku myšlenku převážení kontrabandu přes kontrolovanou síť a zamysleme se nad méně nezákonnou aplikací. Představme si hornatá území severozápadního Vietnamu nacházející se blízko hranic s Laosem a Čínou. Dále si představme, že údolí jsou hluboká a obklopující srázy jsou tak strmé, že je potřeba některá údolí překlenout mosty a lávkami. Po proběhlé vichřici a silných deštích byly některé mosty a lávky zničeny. Ve vesnici se nachází mnoho více či méně zraněných, které je třeba dopravit přes síť vesnic do malého města s nemocnicí.

Někteří zdatní a méně zranění vesničané přichází s nabídkou pomoci. Jsou solidární s více zraněnými a uvědomují si, že jejich zranění nejsou příliš závažná. Vědí, které mosty se zřítily a přes které, pokud by se je podařilo obnovit, by bylo možné přenést nejvíce vážně zraněných.

Naše vesnice pro nás bude počátečním bodem s a malé město s nemocnicí pro nás bude cílovým městem t. Budeme uvažovat síť a scénáře jako v podkapitole 6.1. Předpokládejme, že oprava libovolného zříceného mostu nebo lávky si vyžádá práci 12 vesničanů. Problém lze modelovat následujícím způsobem

$$\max \sum_{r\in\mathcal{S}} p_r\big(x^r_{ts} - \sum_{(i,j)\in A} d_{ij}\phi^r_{ij}\big),$$

za podmínek

$$\sum_{j\in N'} x^r_{sj} - \sum_{i\in N'} x^r_{is} - x^r_{ts} \le 0, \tag{6.7}$$

$$\sum_{j\in N} x^r_{nj} - \sum_{i\in N} x^r_{in} \le 0, \qquad \forall n \in N', \tag{6.8}$$

$$\sum_{j\in N'} x^r_{tj} - \sum_{i\in N'} x^r_{it} + x^r_{ts} \le 0, \tag{6.9}$$

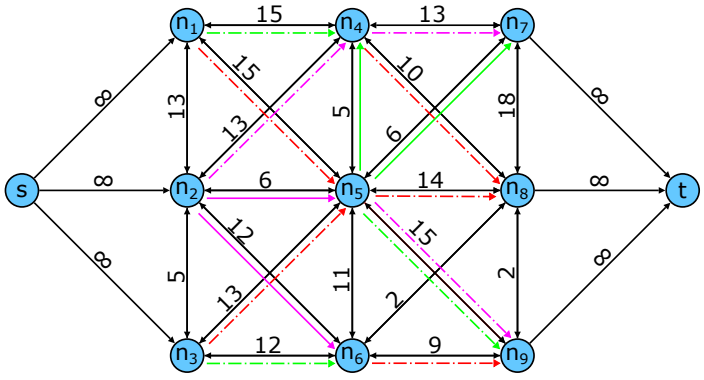
$$0 \le x^r_{ij} \le u^r_{ij} + u_{ij}\phi^r_{ij}, \qquad \forall (i,j) \in A, \tag{6.10}$$

$$x^1_{ij} = x^r_{ij}, \qquad \forall r \in \mathcal{S}. \tag{6.11}$$

kde

$$\phi^r_{ij} = \begin{cases} \in \{0,1\} & \text{pokud je hrana zničena, a tedy } u^r_{ij} = 0, \\ 0 & \text{pokud hrana není zasažena, a tedy } u^r_{ij} = u_{ij}, \end{cases}$$

kde u_{ij} je původní kapacita hrany, viz čísla na hranách na obrázku 6.6. Maximalizujeme účelovou funkci přes všechny scénáře. Původní účelová funkce je nyní doplněna o kompenzaci. Účelová funkce tedy vyjadřuje počet přepravených zraněných redukovaný o počet vesničanů, kteří obětovali ošetření v nemocnici výměnou za to, že pomohou s opravami mostů. První tři omezení (6.7) - (6.9) jsou toková omezení užívaná již dříve. Omezení (6.10) je kapacitní omezení, které nám díky definici proměnné ϕ^r_{ij} říká, jestliže most není při vichřici zničen, není třeba ho opravit a jeho kapacita zůstává beze změny. Pokud je most při vichřici zničen, pak ho lze opravit a obnovit tak jeho původní kapacitu. Za toto rozhodnutí ovšem musíme obětovat ošetření některých vesničanů, kteří pomohou s opravami. Poslední omezení (6.11) je neanticipativní omezení říkající, že pro všechny scénáře budeme rozhodovat stejným způsobem. Pravděpodobnost scénáře p_r je pro všechna $r \in \mathcal{S}$ stejná a rovná se $\frac{1}{|\mathcal{S}|}$. Parametr d_{ij} je pro všechny hrany stejný a vyjadřuje, kolik vesničanů musí spojit síly, aby mohli opravit most. Využijeme modelovou síť z podkapitoly 6.1.



Obrázek 6.6: Modelová úloha se scénáři z podkapitoly 6.1.

Každá barva reprezentuje scénář, který jsme uvedli v podkapitole 6.1. Pokud je čára barevné hrany čerchovaná, znamená to, že se vesničané rozhodli tuto hranu pro příslušný scénář opravit. Výsledky uvedeme formou výstupního .lst souboru získaného z programu *Twostage.gms*.

```
----     98 VARIABLE phi.L

                1           2           3

n1.n4       1.000
n1.n5                   1.000
n2.n4                               1.000
n3.n5                   1.000
n3.n6       1.000
n4.n7                               1.000
n4.n8                   1.000
n5.n7       1.000
n5.n8                   1.000
n5.n9       1.000                   1.000
n6.n9                   1.000

----     98 VARIABLE z.L                    =       20.000
```

Proměnná `phi.L` ve výpisu z programu GAMS odpovídá proměnné ϕ_{ij}^r, kde každý sloupec ve výpisu značí opravené hrany pro příslušný scénář. Optimální hodnota účelové funkce je pak vyjádřena proměnnou `z.L`. Dvoustupňová úloha byla obdobně jako předchozí testována pro větší sítě a větší počty scénářů. Program *Twostage10x30.gms* obsahuje model s řešením pro síť s 30 vnitřními uzly a pro 10 scénářů, program *Twostage30x100.gms* pak pro síť se 100 vnitřními uzly a 30 scénáři.

7. Závěr

Ve své diplomové práci jsem se zabýval modely napadení sítě a jejich implementací v programovacím jazyce GAMS. Po úvodní kapitole, která shrnuje historické souvislosti a uvádí některé důvody, proč má smysl se problematikou zabývat, následují dvě kapitoly poskytující teoretický základ.

Druhá kapitola obsahuje základní pojmy z teorie grafů včetně obrázků ilustrujících některé pojmy. Důležitost této kapitoly nespočívá pouze v možnosti srozumitelně a ilustrativně interpretovat síťové úlohy, ale také v některých uvedených větách, ze kterých vychází způsob modelování napadení sítě a způsob hledání řešení.

Třetí kapitola obsahuje vybrané věty a definice z matematického programování, na které navazují pojmy vztahující se k lineárnímu programování. Na tyto dvě zmíněné podkapitoly navazuje úloha o maximálním toku a na příkladu odvozená duální úloha. Obě úlohy jsou při úvahách o napadení sítě využívány v následující podkapitole. Úloha napadení sítě je dvouúrovňová úloha, která má dva rozhodovatele s odlišnými zájmy. Způsob modelování napadení sítě jako dvouúrovňové úlohy spočívá v propojení primární a duální úlohy tak, jak bylo ukázáno. Zároveň se podařilo dvouúrovňovou úlohu rozdělit na dvě samostatné úlohy - úlohu napadení sítě a navazující úlohu o maximálním toku. Rozdělení úloh se využívá ve čtvrté a páté kapitole.

Jako čtvrtá v pořadí je uvedena kapitola týkající se stochastického programování včetně uvedených deterministických reformulací, které jsme využívali v šesté kapitole. Na deterministické reformulace jsme navázali lineárními dvoustupňovými úlohami s kompenzací, které jsme využili v závěru šesté kapitoly, kde jsme umožnili znovuotevření některých hran.

V páté kapitole jsme se zabývali deterministickými modely napadení sítě. Pro napadení sítě jsme díky předchozímu odvození mohli využívat jednu část dvouúrovňové úlohy a získané parametry pak využít při hledání maximálního toku v porušené síti. V kapitole je obsaženo celkem šest modelů, které nastiňují možnosti napadení sítě formou přerušování hran, uzlů či obojího se sdílenými i nesdílenými zdroji pro přerušování. Další možnosti napadení, jimiž by bylo možné se zabývat, představuje například konkurenční boj autobusových dopravců a jejich potenciální snaha získat převahu na kompletních trasách jednotlivých linek spojů. Ke všem modelům uvedeným v této kapitole náleží vlastní program napsaný v programovacím jazyce GAMS, který je součástí přílohy.

Šestá kapitola zahrnuje stochastické případy napadení sítě. Využívali jsme dříve uvedené deterministické reformulace. Na problémy stochastického napadení sítě se lze dívat jako na dílo přírody, která zasahuje nahodile a nemá za primární cíl škodit našim zájmům. Další směr pohledu nabízí iracionální útočník, jehož útoky vykazují nahodilé chování, nebo racionální útočník, jehož další cíle a záměry nám nejsou známy. Navázali jsme lineární dvoustupňovou úlohou, kde jsme povolili otevření hran za cenu kompenzace snižující hodnotu účelové funkce. Všechny modely v šesté kapitole byly zároveň testovány na větších sítích, aby se ověřila jejich funkčnost a použitelnost. Všechny modely jsou zpracovány jako vlastní programy v GAMSu a jsou uvedeny v příloze.

Programy zpracované v programovacím jazyce GAMS jsou součásti přílohy na přiloženém CD. Modely jsme uvažovali na malé modelové síti. Výpočty byly provedeny také na větších náhodně generovaných sítích, které jsou pro svůj rozsah taktéž součástí přiloženého CD. V případě reálných rozsáhlejších síti, komplikovanějších modelů s větším počtem omezení a třeba i nelineární účelovou funkcí by bylo vhodné do modelování této

problematiky také zahrnout dekompoziční algoritmy, které umožní nejen vypořádání se s nelinearitami, ale zároveň také urychlí hledání optimálního řešení.

Závěrem je vhodné uvést, zda existuje i jiný způsob modelování problémů napadení sítě nebo podobných úloh. Další potenciální způsob, jak modelovat tuto problematiku představují Stackelbergovy hry patřící do oblasti matematiky zvané teorie her. Často se v takovém případě uvažují sekvenční hry dvou hráčů, kdy hráči hrají postupně jeden po druhém. Byť nebylo záměrem tento přístup v práci rozvíjet, byla by škoda ho alespoň nezmínit, protože teorie her představuje zajímavou oblast matematiky vhodnou pro tyto aplikace.

Literatura

[1] AGNETIS, A. *The dual of the maximum flow* [online]. Siena: Universita di Siena, Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, [cit. 2018-04-04]. Dostupné z: http://www.dii.unisi.it/~agnetis/mincutENG.pdf.

[2] BAZARAA, M. S., JARVIS, J. J., SHERALI, H. D. *Linear programming and network flows*. 4. vydání. Hoboken, New Jersey: John Wiley, 2010. ISBN 978-0-470-46272-0.

[3] BAZARAA, M. S., SHERALI, H. D., SHETTY, C. M. *Nonlinear programming: Theory and algorithms*. 3. vydání. Hoboken, New Jersey: John Wiley, 2006. ISBN 978-0-471-48600-8.

[4] BIRGE, J. R., LOUVEAUX, F. *Introduction to stochastic programming*. 2. vydání. New York: Springer, 2011. ISBN 978-1-4614-0236-7.

[5] CABALKA, M. *Celočíselná optimalizace pro řešení dopravních úloh*, bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016.

[6] ČECHOVÁ, D. *Network interdiction*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2006.

[7] DANTZIG, G. B., THAPA, M. N. *Linear programming: Introduction*. New York: Springer, 1997. ISBN 0-387-94833-3.

[8] DANTZIG, G. B., THAPA, M. N. *Linear programming: Theory and extensions*. New York: Springer, 1997. ISBN 0-387-98613-8.

[9] DEMEL, J. *Grafy a jejich aplikace*. Vydání 1. Praha: Academia, 2002. 258 s. ISBN 80-200-0990-6.

[10] FORD, L.R., FULKERSON D.R. *Flows in networks*. Santa Monica, California: Rand Corporation, 1962. ISBN 0-691-14667-5.

[11] *GAMS Documentation 25.0*. [cit. 13-05-2018]. Dostupné z: https://www.gams.com/latest/docs/.

[12] GHIANI, G., LAPORTE, G., MUSMANNO, R. *Introduction to logistics systems planning and control*. Hoboken, New Jersey: John Wiley, 2004. ISBN 0-470-84917-7.

[13] GRIVA, I., NASH, S. G., SOFER A. *Linear and nonlinear optimization*. 2. vydání. Philadelphia: Society for Industrial and Applied Mathematics, c2009. ISBN 978-0-898716-61-0.

[14] HELMBOLD, R.L. *A counter capacity network interdiction model*. Santa Monica, California: Rand Corporation, 1971.

[15] HOLEŠOVSKÝ, J. *Traffic assignment optimization models*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2012.

[16] HRABEC, D. *Modely stochastického programování pro inženýrský návrh*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2011.

[17] KALL, P., WALLACE, S. W. *Stochastic programming*. New York: Wiley, 1995. ISBN 0-471-95158-7.

[18] KENNEDY, K. T., DECKRO, R. F., MOORE, J. T., HOPKINSON, K. M. *Nodal Interdiction* [online]. Mathematical and Computer Modelling. Elsevier, 2011, **54**(11-12), 3116-3125 [cit. 2018-02-19]. DOI: 10.1016/j.mcm.2011.07.041. ISSN 08957177. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0895717711004675?via%3Dihub.

[19] KLAPKA, J., DVOŘÁK J., POPELA, P. *Metody operačního výzkumu*. Vydání 1. Brno: Vysoké učení technické v Brně, 1996. 154 s. ISBN 80-214-0817-0.

[20] KLIMEŠ, L. *Stochastic programming algorithms*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2010.

[21] LESSARD, L. *Dual flows and algorithms* Prezentace k předmětu „Introduction to optimization“ na University of Wisconsin-Madison, r. 2017, [cit. 2018-04-02]. Dostupné z: http://www.laurentlessard.com/teaching/cs524/slides/7%20-%20dual%20flows%20and%20algorithms.pdf.

[22] MÁLEK, M. *Stochastic optimization of network flows*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017.

[23] MIKULÁŠEK, K. *Grafy a algoritmy* [online]. Soubor přednášek na ÚM FSI VUT v Brně, r. 2009, [cit. 2018-02-25]. Dostupné z: http://teaching.mikulasek.net/Lectures/graphs.html.

[24] MORTON, D. P. *Stochastic network interdiction* [online]. Hoboken, New Jersey: John Wiley, 2011, [cit. 2018-04-06] ISBN 9780470400531. Dostupné z: https://onlinelibrary.wiley.com/doi/10.1002/9780470400531.eorms0835.

[25] MORTON, D. P., PAN, F., SAEGER, K. J. *Models for nuclear smuggling interdiction* [online]. IIE Transactions. 2007, **39**(1), 3-14. DOI: 10.1080/07408170500488956. ISSN 0740-817X. Dostupné z: http://www.tandfonline.com/doi/abs/10.1080/07408170500488956.

[26] NEŠETŘIL, J. *Teorie grafů*. Vydání 1. Praha: SNTL, 1979, 320 s. ISBN 04-017-79.

[27] NEVRLÝ, V. *Modely a metody pro svozové úlohy*, diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016.

[28] PLESNÍK, J. *Grafové algoritmy*. Bratislava: VEDA, 1983.

[29] POPELA, P. *An objected-oriented approach to multistage stochastic programming*, disertační práce. Praha: Univerzita Karlova, 1998.

[30] POPELA, P. *Optimization I* [online]. Přednáškový text pro předmět Optimalizace I na ÚM FSI VUT v Brně, r. 2018, [cit. 2018-04-07]. Dostupné z: http://math.fme.vutbr.cz/cz/zamestnanci.

[31] STEINRAUF, R. L. *Network interdiction models*. Monterey, California: Naval postgraduate school, 1991.

[32] RUSZCZYNSKI, A. et al. *Handbooks in operations research and management science, vol. 10: Stochastic programming*. Amsterdam: Elsevier, 2003. ISBN 978-0-444-50854-6.

[33] TRAVES, W. *The network interdiction problem* [online]. Annapolis: United States Naval Academy, Department of Mathematics, 2012. [cit. 2018-01-18] Dostupné z: https://www.usna.edu/Users/math/traves/papers/interdiction.pdf.

[34] WALLACE, S.W., KING, A. *Modeling with stochastic programming*. Springer: 2012. ISBN 978-0-387-87816-4.

[35] WOLLMER, R.D. *Algorithms for targeting strikes in a lines of communication network*[online]. Operations Research, 1970, **18**(3), 497-515 [cit. 2018-04-05]. ISSN 0030-364X. Dostupné z: http://web.b.ebscohost.com.ezproxy.lib.vutbr.cz/ehost/detail/detail?vid=0&sid=5abaf1cd-bdd4-4f39-8fa3-9473a4687c02%40sessionmgr120&bdata=Jmxhbmc9Y3Mmc2l0ZT1laG9zdC1saXZl#db=bth&AN=8604749

[36] WOLSEY, L.A. *Integer programming*. New York: John Wiley & Sons, 1998. ISBN 978-0-4-1-28366-9.

[37] WOOD, R. K. *Deterministic network interdiction* [online]. Mathematical and Computer Modelling. Elsevier, 1993, **17**(2), 1-18 [cit. 2018-02-10]. DOI: 10.1016/0895-7177(93)90236-R. ISSN 08957177. Dostupné z: http://www.sciencedirect.com/science/article/pii/089571779390236R.

8. Seznam příloh

8.1. Seznam přiložených souborů

Data jsou rozdělena do adresářů podle kapitol, v nichž se vyskytují.

1. Deterministické modely napadení sítě (kapitola 5)
   1.1. Model pro přerušování hran (5.1)
      - `BDM.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.1.
   1.2. Model pro přerušování hran s více zdroji (5.2)
      - `EBDM2.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.2.
   1.3. Model pro přerušování hran s více zdroji, zdrojovými a cílovými uzly (5.3)
      - `Gen.m` - soubor pro generování testovacích dat z programu Matlab.
      - `Prerusovani_hran.xlsx` - soubor s daty využívanými pro testovací úlohu.
      - `EBDM3.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.3.
   1.4. Model pro přerušování uzlů (5.4)
      - `Nodal_deletion.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.4.
   1.5. Model pro přerušování uzlů a hran se sdílenými zdroji (5.5)
      - `Nodal_deletion_shared.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.5.
   1.6. Model pro přerušování uzlů a hran s nesdílenými zdroji (5.6)
      - `Nodal_deletion_different.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 5.6.
2. Stochastické modely napadení sítě (kapitola 6)
   2.1. WS přístup k napadení sítě (6.1)
      - `WS.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 6.1.
      - `WS10x30.gms` - zdrojový kód v GAMSu pro WS přístup k napadení sítě pro 10 scénářů a 30 vnitřních uzlů.
      - `WS30x100.gms` - zdrojový kód v GAMSu pro WS přístup k napadení sítě pro 30 scénářů a 100 vnitřních uzlů.
   2.2. HN přístup k napadení sítě (6.2)
      2.2.1. IS přístup k napadení sítě (6.2.1)
         - `IS.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 6.2.1.
         - `IS10x30.gms` - zdrojový kód v GAMSu pro IS přístup k napadení sítě pro 10 scénářů a 30 vnitřních uzlů.
         - `IS30x100.gms` - zdrojový kód v GAMSu pro IS přístup k napadení sítě pro 30 scénářů a 100 vnitřních uzlů.
      2.2.2. EV přístup k napadení sítě (6.2.2)

- `EV.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 6.2.2.
- `EV10x30.gms` - zdrojový kód v GAMSu pro EV přístup k napadení sítě pro 10 scénářů a 30 vnitřních uzlů.
- `EV30x100.gms` - zdrojový kód v GAMSu pro EV přístup k napadení sítě pro 30 scénářů a 100 vnitřních uzlů.

2.2.3. MM přístup k napadení sítě (6.2.3)

- `MM.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 6.2.3.
- `MM10x30.gms` - zdrojový kód v GAMSu pro MM přístup k napadení sítě pro 10 scénářů a 30 vnitřních uzlů.
- `MM30x100.gms` - zdrojový kód v GAMSu pro MM přístup k napadení sítě pro 30 scénářů a 100 vnitřních uzlů.

2.3. Dvoustupňové napadení sítě s kompenzací (6.3)

- `Twostage.gms` - zdrojový kód v GAMSu k modelu z podkapitoly 6.3.
- `Twostage10x30.gms` - zdrojový kód v GAMSu pro dvoustupňovou úlohu napadení sítě pro 10 scénářů a 30 vnitřních uzlů.
- `Twostage30x100.gms` - zdrojový kód v GAMSu pro dvoustupňovou úlohu napadení sítě pro 30 scénářů a 100 vnitřních uzlů.

8.2. GAMS

V této části jsou uvedeny zdrojové kódy využité při výpočtech. Pro modely z kapitoly 5 se jedná o modely vztahující se k modelové síti. V případě stochastických modelů z kapitoly 6 jsou vybrány modely pro generované sítě, neboť pro modelovou síť z textu byla data složitě zobrazitelná.

Model pro přerušování hran

```
$Ontext
Prvni model pro prerusovani hran pri napadani site vychazi z modelu zmineneho
v clanku Deterministic Network Inderdiction Problem od Kevina Wooda. Data pro
tento priklad byla vygenerovana v programu Matlab. Predpoklada se, ze utocnik
je inteligentni, ma dostatecne znalosti site, a proto se snazi sit napadnout
co nejefektivneji vzhledem ke svemu budgetu a cenam za preruseni jednotlivych
hran. Kapacity prerusenych hran jsou odecteny od puvodni kapacitni matice.
Nove vznikla matice je pouzita pro nasledujici optimalizacni ulohu pro uzivatele
site, ktery se v ponicene siti snazi maximalizovat mozny tok. Znaceni je prevzato
ze zmineneho clanku.
$Offtext

*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/;

Alias (i,j);

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
Table u(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     13    0     15    15    0     0     0     0     0
n2  0    13    0     5     13    6     12    0     0     0     0
n3  0    0     5     0     0     13    12    0     0     0     0
n4  0    15    13    0     0     5     0     13    10    0     0
```

```
n5  0    15    6     13    5     0     11    6     14    15    0
n6  0    0     12    12    0     11    0     0     2     9     0
n7  0    0     0     0     13    6     0     0     18    0     1000
n8  0    0     0     0     10    14    2     18    0     2     1000
n9  0    0     0     0     0     15    9     0     2     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Matice cen za preruseni jednotlivych hran. Pokud je hodnota =  0, hrana neexistuje.
Table c(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     1     0     1     1     0     0     0     0     0
n2  0    1     0     1     1     1     1     0     0     0     0
n3  0    0     1     0     0     1     1     0     0     0     0
n4  0    1     1     0     0     1     0     1     1     0     0
n5  0    1     1     1     1     0     1     1     1     1     0
n6  0    0     1     1     0     1     0     0     1     1     0
n7  0    0     0     0     1     1     0     0     1     0     1000
n8  0    0     0     0     1     1     1     1     0     1     1000
n9  0    0     0     0     0     1     1     0     1     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Utocnikuv budget.
Parameter
budget / 3 /;

$Ontext
Zavedeni binarnich promennych. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t. Gamma(i,j) = 1, pokud hrana nalezi
do minimaniho rezu a je prerusena, jinak = 0. Beta(i,j) = 1, pokud hrana nalezi
do minimalniho rezu, ale neni prerusena, jinak = 0.
$Offtext

Binary variable
alpha(i)
gamma(i,j)
beta(i,j);

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
x(i,j);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
z1
z2
iflow(i,j)
util(i,j);

*Napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1       omezeni pro pocatecni uzel
          stc2       omezeni pro cilovy uzel
          bud        utocnikuv budget
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + gamma(i,j) =g= 0;
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud.. sum((i,j), c(i,j) * gamma(i,j)) =l= budget;
```

```
iflw(i,j).. iflow(i,j) =e= u(i,j) * gamma(i,j);
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Model pro maximalni tok v residualni siti ponicene utocnikem.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);

model BDMattack /obj1, bal1, stc1, stc2, bud, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, gamma.l, iflow.l, x.l, z2.l;
```

Model pro přerušování hran s více zdroji

```
$ontext
Nasledujici model navazuje na model v programu BDM.gms. Predpoklada se, ze utocnik
potrebuje pro preruseni hrany vice surovin najednou. Ma tedy dva ruzne budgety.
Navic se predpoklada, ze pokud pouzije potrebne mnozstvi pouze jedne suroviny
muze hranu znicit pouze z jedne poloviny. (V pripade trech pozadovanych surovin
z jedne tretiny, atd.). Navic se zde predpoklada, ze hranu lze znicit take castecne.
$offtext

*Zavedeni mnozin uzlu a surovin, podmnoziny bez startovacicho a ciloveho uzle.
Set
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/
k /1*2/;

Alias (i,j);

*Nastaveny budget utocnika.
Parameter budget(k) / 1 10, 2 25/

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
Table u(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     13    0     15    15    0     0     0     0     0
n2  0    13    0     5     13    6     12    0     0     0     0
n3  0    0     5     0     0     13    12    0     0     0     0
n4  0    15    13    0     0     5     0     13    10    0     0
n5  0    15    6     13    5     0     11    6     14    15    0
n6  0    0     12    12    0     11    0     0     2     9     0
n7  0    0     0     0     13    6     0     0     18    0     1000
n8  0    0     0     0     10    14    2     18    0     2     1000
n9  0    0     0     0     0     15    9     0     2     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Cena v ruznych surovinach za preruseni jednotlivych hran.
Table c(i,j,k)
   s.1  n1.1 n2.1 n3.1 n4.1 n5.1 n6.1 n7.1 n8.1 n9.1 t.1  s.2 n1.2 n2.2 n3.2 n4.2 n5.2 n6.2 n7.2 n8.2 n9.2 t.2
s   0    1000 1000 1000 0    0    0    0    0    0    0    0   1000 1000 1000 0    0    0    0    0    0    0
n1  0    0    3    0    4    3    0    0    0    0    0    0   0    6    0    9    9    0    0    0    0    0
n2  0    3    0    4    3    3    3    0    0    0    0    0   6    0    7    9    11   6    0    0    0    0
n3  0    0    4    0    0    2    4    0    0    0    0    0   0    7    0    0    4    11   0    0    0    0
n4  0    4    3    0    0    2    0    3    3    0    0    0   9    9    0    0    7    0    14   10   0    0
n5  0    3    3    2    2    0    3    3    3    1    0    0   9    11   4    7    0    8    3    10   12   0
n6  0    0    3    4    0    3    0    0    2    3    0    0   0    6    11   0    8    0    0    8    13   0
n7  0    0    0    0    3    3    0    0    3    0    1000 0   0    0    0    14   3    0    0    9    0    1000
n8  0    0    0    0    3    3    2    3    0    1    1000 0   0    0    0    10   10   8    9    0    9    1000
```

```
n9  0    0    0    0    0    1    3    0    1    0    1000 0   0    0    0    0    12   13   0    9    0    1000
t   1000 0    0    0    0    0    0    0    0    0    0    0   0    0    0    0    0    0    0    0    0    0;

$Ontext
Zavedeni binarni promenne. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t.
$Offtext

Binary variable
alpha(i);

$Ontext
Oproti modelu v programu BDM.gms zde gamma(i,j,k) a beta(i,j) vystupuji jako
nezaporne promenne. Gamma(i,j,k) vyjadruje miru poskozeni hrany (i,j) vzhledem
k pouzite surovine 'k'. Pokud je Gamma(i,j,k) = 1, hrana je zcela z nicena
z 1/k casti pomoci suroviny 'k'. Beta(i,j) vyjadruje 'miru nezniceni' hrany (i,j),
tj. 1 pokud hrana neni znicena, 0.5 pokud je hrana znicena z jedne poloviny apod.
Promenna x(i,j) opet vyjadruje nezaporny tok na hrane (i,j).
$Offtext

Positive variable
x(i,j)
beta(i,j)
gamma(i,j,k);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
z1
z2
iflow
util(i,j);

$Ontext
Pro uplne preruseni hrany je treba, aby Gamma(i,j,k) = 1 pro vsechna k.
$Offtext

*Napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1       omezeni pro pocatecni uzel
          stc2       omezeni pro cilovy uzel
          bud(k)     utocnikuv budget
          yc1        dolni mez pro hodnotu zniceni gamma
          yc2        horin mez pro hodnotu zniceni gamma
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice pro napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + sum(k,gamma(i,j,k)/card(k)) =g= 0;
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud(k).. sum((i,j), c(i,j,k) * gamma(i,j,k)) =l= budget(k);
yc1(i,j,k).. 0 =l= gamma(i,j,k);
yc2(i,j,k).. gamma(i,j,k) =l= 1;
iflw(i,j).. iflow(i,j) =e= u(i,j) * sum(k,gamma(i,j,k)/card(k));
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Model pro maximalni tok v residualni siti ponicene utocnikem.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
```

```
          xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s') =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s') =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);

model BDMattack /obj1, bal1, stc1, stc2, bud, yc1, yc2, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, gamma.l, iflow.l, x.l, z2.l;
```

Model pro přerušování hran s více zdroji, zdrojovými a cílovými uzly

```
$ontext
Posledni model urceny pouze pro prerusovani hran. Predpoklada se oproti modelu
z programu EBDM2.gms, ze mame vice zdrojovych a vice cilovych uzlu. Zaroven
zustala vlastnost modelu, diky ktere je pro preruseni hran potreba vice surovin,
navic hrany jdou znicit i castecne, stejnym zpusobem jako v predchozim modelu.
Oproti predchozim modelu je zde navic pridana cast kodu pro nacitani dat z Excelu
poskytujici vetsi komfort a obratnost pri pouzivani programu.
$offtext

*Preddefinovany mnozin a podmnozin uzlu, ktere budou nacteny z Exceloveho souboru.
Set
i mnozina vsech uzlu
n(i) podmnozina uzlu bez zdrojovych a cilovych uzlu
s(i) mnozina zdrojovych uzlu bez super-zdroje
t(i) set of sinks without super-sink
k mnozina surovin potrebna k preruseni hran /1*2/  ;

Alias (i,j);

*Preddefinovani nacitanych parametru a zadefinovani budgetu utocnika.
Parameter u(i,j) kapacity hran
          c(i,j,k) suroviny potrebne k preruseni hran
          budget(k) utocnikuv budget /1 12, 2 20/;

*Nacitain mnozin a podmnozin z prilozeneho Exceloveho souboru.
$onecho > tasks3.txt
set=i rng=Indices!A3 rdim=1 set=n rng=Indices!C3 rdim=1 set=s rng=Indices!E3
rdim=1 set=t rng=Indices!G3 rdim=1 par=u rng=Capacity!B3:R19
$offecho

$call gdxxrw.exe ebdm1input3_final.xlsx trace=3 @tasks3.txt
$gdxin ebdm1input3_final.gdx
$loaddc i n s t u
$gdxin

$Ontext
Nacitani vicerozmerne matice c(i,j,k) vyzadujici specialni proceduru, ktera
je uvedena nize.
$Offtext

$onecho > tasks31.txt
I="ebdm1input3_final.xlsx"
R=PaC!B3:AH19
O=tasks31.inc
$offecho
$call =xls2gms @tasks31.txt
Table c(i,j,k)
$include tasks31.inc
;

$Ontext
Zavedeni binarni promenne. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t.
```

```
$Offtext
Binary variable
alpha(i);

$Ontext
Oproti modelu v programu BDM.gms zde gamma(i,j,k) a beta(i,j) vystupuji jako
nezaporne promenne. Gamma(i,j,k) vyjadruje miru poskozeni hrany (i,j) vzhledem
k pouzite surovine 'k', pokud nalezi do minimalniho rezu. Pokud je Gamma(i,j,k) = 1,
hrana je zcela z nicena z 1/k casti pomoci suroviny 'k'. Beta(i,j) vyjadruje
'miru nezniceni' hrany (i,j), tj. = 1 pokud hrana neni znicena a pritom patri
do minimalniho rezu, 0.5 pokud je hrana znicena z jedne poloviny apod.
Promenna x(i,j) opet vyjadruje nezaporny tok na hrane (i,j).
$Offtext
Positive variable
x(i,j)
beta(i,j)
gamma(i,j,k);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
z1
z2
iflow
util(i,j);

$Ontext
Pro uplne preruseni hrany je treba, aby Gamma(i,j,k) = 1 pro vsechna k.
$Offtext

*Napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1(s)    omezeni pro zdrojovy uzel
          stc2(t)    omezeni pri cilovy uzel
          bud(k)     utocnikuv budget
          yc1(i,j,k) dolni mez pro hodnotu zniceni gamma
          yc2(i,j,k) dolni mez pro hodnotu zniceni gamma
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + sum(k,gamma(i,j,k)/card(k)) =g= 0;
stc1(s).. alpha(s) =e= 0;
stc2(t).. alpha(t) =e= 1;
bud(k).. sum((i,j), c(i,j,k) * gamma(i,j,k)) =l= budget(k);
yc1(i,j,k).. 0 =l= gamma(i,j,k);
yc2(i,j,k).. gamma(i,j,k) =l= 1;
iflw(i,j).. iflow(i,j) =e= u(i,j) * sum(k,gamma(i,j,k)/card(k));
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Model pro maximalni tok v residualni siti ponicene utocnikem.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('T','S');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('S',n)) - sum(n,x(n,'S')) - x('T','S') =l= 0;
ter2.. sum(n,x('T',n)) - sum(n,x(n,'T')) + x('T','S') =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);
```

```
model BDMattack /obj1, bal1, stc1, stc2, bud, yc1, yc2, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, gamma.l, iflow.l, x.l, z2.l;
```

Model pro přerušování uzlů

```
$Ontext
Model prevzaty z clanku Nodal Interdiction od autoru Deckra, Kennedyho, Moora
a Hopkinsona umoznuje prerusovat uzly. Preruseni uzlu v tomto pripade znamena
i preruseni vsech z uzlu vystupujicich hran. Uzel se tak tedy stava slepou ulickou,
kterou v nasledujicim problemu o maximalnim toku uzivatel site nemuze vyuzit.
$Offtext

*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/;

Alias (i,j);

*Kapacitni matice jednotlivych hran.
Table u(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     13    0     15    15    0     0     0     0     0
n2  0    13    0     5     13    6     12    0     0     0     0
n3  0    0     5     0     0     13    12    0     0     0     0
n4  0    15    13    0     0     5     0     13    10    0     0
n5  0    15    6     13    5     0     11    6     14    15    0
n6  0    0     12    12    0     11    0     0     2     9     0
n7  0    0     0     0     13    6     0     0     18    0     1000
n8  0    0     0     0     10    14    2     18    0     2     1000
n9  0    0     0     0     0     15    9     0     2     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Utocnikuv budget a vektor cen za preruseni jednotlivych uzlu.
Parameter
budget / 2 /
c(i) /s 1000, n1 1, n2 1, n3 1, n4 1, n5 1, n6 1, n7 1, n8 1, n9 1, t 1000/ ;

$Ontext
Zavedeni binarnich promennych. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t. Gamma(i) = 1, pokud je uzel znicen,
jinak = 0. Gamma(i,j) = 1, pokud je znicen uzel i, je s nim znicena i z nej
vystupujici hrana, jinak = 0. Beta(i,j) = 1, pokud hrana nalezi
do minimalniho rezu, ale neni prerusena, jinak = 0.
$Offtext

Binary variable
alpha(i)
beta(i,j)
delta(i)
gamma(i,j);

*X(i,j) reprezentuje tok na hrane (i,j).
Positive variable
x(i,j);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
z1
```

```
z2
iflow
util(i,j)
result;

*Napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1       omezeni pro pocatecni uzel
          stc2       omezeni pro cilovy uzel
          bud        utocnikuv budget
          nec(i,j)   omezeni pro uzel a z nej vychazejici hrany
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + gamma(i,j) =g= 0 $(ord(i) <> ord(j));
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud.. sum(i, c(i) * delta(i)) =l= budget;
nec(i,j)$(u(i,j)>0).. gamma(i,j) =e= delta(i);
iflw(i,j).. iflow(i,j) =e= u(i,j) * gamma(i,j);
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Uloha maximalniho toku pro uzivatele site.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);

model BDMattack /obj1, bal1, stc1, stc2, bud, iflw, iutil, nec/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc/;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, delta.l, gamma.l, iflow.l, x.l, z2.l;
```

Model pro přerušování uzlů a hran se sdílenými zdroji

```
$Ontext
Model prevzaty z clanku Nodal Interdiction od autoru Deckra, Kennedyho, Moora
a Hopkinsona navazuje na model Nodal_deletion.gms. Umoznuje prerusovat uzly a hrany
zaroven, pricemz se pro preruseni pouzivaji stejne zdroje. Hlavni omezeni je navrzeno
tak, aby se cena hran vychazejicich z preruseneho uzlu nepocitala do budgetu,
ale extra zvlast. Preruseni uzlu a tedy i z nej vychazejicich hran stoji utocnika
pouze cenu preruseni uzlu.
$Offtext

*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/;

Alias (i,j);

*Kapacitni matice U a matice cen za preruseni C.
Table u(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     13    0     15    15    0     0     0     0     0
n2  0    13    0     5     13    6     12    0     0     0     0
n3  0    0     5     0     0     13    12    0     0     0     0
n4  0    15    13    0     0     5     0     13    10    0     0
```

```
n5  0    15    6     13    5     0     11    6     14    15    0
n6  0    0     12    12    0     11    0     0     2     9     0
n7  0    0     0     0     13    6     0     0     18    0     1000
n8  0    0     0     0     10    14    2     18    0     2     1000
n9  0    0     0     0     0     15    9     0     2     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

Table c2(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     1     0     1     1     0     0     0     0     0
n2  0    1     0     1     1     1     1     0     0     0     0
n3  0    0     1     0     0     1     1     0     0     0     0
n4  0    1     1     0     0     1     0     1     1     0     0
n5  0    1     1     1     1     0     1     1     1     1     0
n6  0    0     1     1     0     1     0     0     1     1     0
n7  0    0     0     0     1     1     0     0     1     0     1000
n8  0    0     0     0     1     1     1     1     0     1     1000
n9  0    0     0     0     0     1     1     0     1     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Utocnikuv rozpocet a vektor cen za preruseni jednotlivych uzlu.
Parameter
budget / 2 /
c1(i) /s 1000, n1 1, n2 1, n3 1, n4 1, n5 1, n6 1, n7 1, n8 1, n9 1, t 1000/ ;

$Ontext
Zavedeni binarnich promennych. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t. Gamma(i) = 1, pokud je uzel znicen,
jinak = 0. Gamma(i,j) = 1, pokud je znicen uzel i, je s nim znicena i z nej
vystupujici hrana, jinak = 0. Beta(i,j) = 1, pokud hrana nalezi
do minimalniho rezu, ale neni prerusena, jinak = 0.
$Offtext

Binary variable
alpha(i)
beta(i,j)
delta(i)
gamma(i,j);

*X(i,j) reprezentuje tok na hrane (i,j).
Positive variable
x(i,j);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
xts
z1
z2
iflow
util(i,j);

*Napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1       omezeni pro pocatecni uzel
          stc2       omezeni pro cilovy uzel
          bud        utocnikuv budget
          nec(i,j)   omezeni pro uzel a z nej vychazejici hrany
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
```

```
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + gamma(i,j) =g= 0;
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud.. sum((i,j),c2(i,j)*gamma(i,j)) - sum((i,j),c2(i,j)*delta(i)) + sum(i,delta(i)*c1(i)) =l= budget;
nec(i,j)$(u(i,j)>0).. gamma(i,j) =g= delta(i);
iflw(i,j).. iflow(i,j) =e= u(i,j) * gamma(i,j);
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Uloha o maximalnim toku pro uzivatele site.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);

model BDMattack /obj1, bal1, stc1, stc2, bud, nec, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, delta.l, gamma.l, iflow.l, x.l, z2.l;
```

Model pro přerušování uzlů a hran s nesdílenými zdroji

```
$Ontext
Model prevzaty z clanku Nodal Interdiction od autoru Deckra, Kennedyho, Moora
a Hopkinsona navazuje na model Nodal_deletion.gms a Nodal_edge_deletion_shared.gms.
Umoznuje prerusovat uzly a hrany zaroven, pricemz se pro preruseni pouzivaji ruzne
na sobe nezavisle zdroje. Utocnik tedy z pomoci jednoho typu zdroju prerusuje uzly
a s pomoci jineho typu zase hrany.
$Offtext

*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1, n2, n3, n4, n5, n6, n7, n8, n9, t/
n(i) /n1, n2, n3, n4, n5, n6, n7, n8, n9/;

Alias (i,j);

*Kapacitni matice U a matice cen za preruseni C.
Table u(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     13    0     15    15    0     0     0     0     0
n2  0    13    0     5     13    6     12    0     0     0     0
n3  0    0     5     0     0     13    12    0     0     0     0
n4  0    15    13    0     0     5     0     13    10    0     0
n5  0    15    6     13    5     0     11    6     14    15    0
n6  0    0     12    12    0     11    0     0     2     9     0
n7  0    0     0     0     13    6     0     0     18    0     1000
n8  0    0     0     0     10    14    2     18    0     2     1000
n9  0    0     0     0     0     15    9     0     2     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

Table c2(i,j)
    s    n1    n2    n3    n4    n5    n6    n7    n8    n9    t
s   0    1000  1000  1000  0     0     0     0     0     0     0
n1  0    0     1     0     1     1     0     0     0     0     0
n2  0    1     0     1     1     1     1     0     0     0     0
n3  0    0     1     0     0     1     1     0     0     0     0
n4  0    1     1     0     0     1     0     1     1     0     0
n5  0    1     1     1     1     0     1     1     1     1     0
n6  0    0     1     1     0     1     0     0     1     1     0
n7  0    0     0     0     1     1     0     0     1     0     1000
n8  0    0     0     0     1     1     1     1     0     1     1000
```

```
n9  0    0     0     0     0     1     1     0     1     0     1000
t   1000 0     0     0     0     0     0     0     0     0     0;

*Utocnikuv budget1 pro prerusovani uzlu, budget2 pro prerusovani hran a vektor
*cen za preruseni jednotlivych uzlu.
Parameter
budget1 / 1 /
budget2 / 1 /
c1(i) /s 1000, n1 1, n2 1, n3 1, n4 1, n5 1, n6 1, n7 1, n8 1, n9 1, t 1000/ ;

$Ontext
Zavedeni binarnich promennych. Alpha(i) = 1 pro vsechny uzly i na strane
minimalniho rezu blize k cilovemu uzlu t. Gamma(i) = 1, pokud je uzel znicen,
jinak = 0. Gamma(i,j) = 1, pokud je znicen uzel i, je s nim znicena i z nej
vystupujici hrana, nebo pokud je znicena samostatna hrana (i,j), jinak = 0.
Beta(i,j) = 1, pokud hrana nalezi do minimalniho rezu, ale neni prerusena, jinak = 0.
$Offtext

Binary variable
alpha(i)
beta(i,j)
delta(i)
gamma(i,j);

*X(i,j) vyjadruje mnozstvi toku na hrane (i,j).
Positive variable
x(i,j);

$Ontext
Xts je maximalizovane mnozstvi toku v siti, ktere projde z pocatecniho
do cilovehu uzlu. z1 a z2 jsou hodnoty jednotlivych ucelovych funkci.
Iflow(i,j) je pomocna promenna, do niz je zaznamenano mnozstvi toku
preruseneho v prvni uloze na hrane (i,j). Util(i,j) je pomocna promenna
vyjadrujici hodnoty kapacitni matice po odecteni kapacit napadenych hran.
$Offtext

Variable
z1
z2
iflow
util(i,j);

*Model pro napadeni site utocnikem.

Equations obj1       ucelova funkce pro utocnika
          bal1(i,j)  omezeni zachovavajici smer toku v siti
          stc1       omezeni pro pocatecni uzel
          stc2       omezeni pro cilovy uzel
          bud1       utocnikuv budget pro prerusovani uzlu
          bud2       utocnikuv budget pro prerusovani hran
          nec(i,j)   omezeni pro uzel a z nej vychazejici hrany
          iflw(i,j)  preruseny tok
          iutil(i,j) kapacitni matice po napadeni;

obj1.. z1 =e= sum((i,j), u(i,j) * beta(i,j));
bal1(i,j)$(u(i,j)>0)..  alpha(i) - alpha(j) + beta(i,j) + gamma(i,j) =g= 0;
stc1('s').. alpha('s') =e= 0;
stc2('t').. alpha('t') =e= 1;
bud1.. sum(i, c1(i) * delta(i)) =l= budget1;
bud2.. sum((i,j),c2(i,j)*gamma(i,j)) - sum((i,j),c2(i,j)*delta(i)) =l= budget2;
nec(i,j)$(u(i,j)>0).. gamma(i,j) =g= delta(i);
iflw(i,j).. iflow(i,j) =e= u(i,j) * gamma(i,j);
iutil(i,j).. util(i,j) =e= u(i,j) - iflow(i,j);

*Uloha o maximalnim toku uzivatele site.

Equations obj2     ucelova funkce pro uzivatele
          bal2(n)  omezeni zachovavajici tok v uzlu
          src2     omezeni pro tok z pocatecni uzel
          ter2     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;
```

```
obj2.. z2 =e= x('t','s');
bal2(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src2.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter2.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= util.l(i,j);

model BDMattack /obj1, bal1, stc1, stc2, bud1, bud2, nec, iflw, iutil/;
solve BDMattack min z1 using mip;
model BDMutility / obj2, bal2, src2, ter2, xc /;
solve BDMutility max z2 using mip;
display alpha.l, beta.l, delta.l, gamma.l, iflow.l, x.l, z2.l;
```

WS přístup k napadení sítě

```
option optcr=0.0001;
option ResLim = 100000000
option IterLim = 1000000000;
*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1 * n30, t/
n(i) /n1 * n30/
ss / 1*10 /;

Alias (i,j);

Parameter u(i,j)
          gamma(i,j)
          gammas(i,j,ss)
          xs(i,j,ss)
          zws(ss);

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
u(i,j) = uniform(0,1000);
loop((i,j),
if(u(i,j) le 200,u(i,j) = 0;);
if(u(i,j) ge 800,u(i,j) = 1000;););
u('t','s') = 1000000;

*Zasah nahody
gammas(i,j,ss) = uniform(0,1);
loop((i,j,ss),
if(gammas(i,j,ss) le 0.8,gammas(i,j,ss) = 0;);
if(gammas(i,j,ss) gt 0.8,gammas(i,j,ss) = 1;); );

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
x(i,j);

Variable
z;

*Model pro maximalni tok v residualni siti ponicene utocnikem.
Equations obj     ucelova funkce pro uzivatele
          bal(n)  omezeni zachovavajici tok v uzlu
          src     omezeni pro tok z pocatecni uzel
          ter     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj.. z =e= x('t','s');
bal(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= u(i,j)*(1-gamma(i,j));

*=== WAIT-AND-SEE===
model WS / obj, bal, src, ter, xc /;
loop(ss,
gamma(i,j) = gammas(i,j,ss);
solve WS max z using mip;
xs(i,j,ss) = x.l(i,j);
```

```
zws(ss) = z.l
display z.l);
display ss, gammas, xs, zws;
```

IS přístup k napadení sítě

```
option optcr=0.0001;
option ResLim = 100000000
option IterLim = 1000000000;
*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1 * n30, t/
n(i) /n1 * n30/
ss / 1*10 /;

Alias (i,j);

Parameter u(i,j)
          gamma(i,j)
          gammas(i,j,ss)
          xs(i,j,ss);

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
u(i,j) = uniform(0,1000);
loop((i,j),
if(u(i,j) le 50,u(i,j) = 0;);
if(u(i,j) ge 800,u(i,j) = 1000;););
u('t','s') = 1000000;

gammas(i,j,ss) = uniform(0,1);
loop((i,j,ss),
if(gammas(i,j,ss) le 0.9999,gammas(i,j,ss) = 0;);
if(gammas(i,j,ss) gt 0.9999,gammas(i,j,ss) = 1;);  );

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
x(i,j);

Variable
z;

*Model pro maximalni tok v residualni siti ponicene utocnikem.
Equations obj     ucelova funkce pro uzivatele
          bal(n)  omezeni zachovavajici tok v uzlu
          src     omezeni pro tok z pocatecni uzel
          ter     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj.. z =e= x('t','s');
bal(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= u(i,j)*(1-gamma(i,j));

*=== INDIVIDUAL SCENARIO===
model IS / obj, bal, src, ter, xc /;
gamma(i,j) = gammas(i,j,'1');
solve IS max z using mip;
xs(i,j,'1') = x.l(i,j);

loop(ss,
gamma(i,j) = gammas(i,j,ss);
x.lo(i,j) = xs(i,j,'1');
x.up(i,j) = xs(i,j,'1');
solve IS max z using mip);
display  x.l, gamma, z.l;
```

EV přístup k napadení sítě

```
option optcr=0.0001;
```

```
option ResLim = 100000000
option IterLim = 1000000000;
*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1 * n30, t/
n(i) /n1 * n30/
ss / 1*10 /;

Alias (i,j);

Parameter u(i,j)
          gamma(i,j)
          gammas(i,j,ss)
          xs(i,j,ss)
          zs(ss)
          zev;

Parameter p(ss);
p(ss) = 1/card(ss);
*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
u(i,j) = uniform(0,1000);
loop((i,j),
if(u(i,j) le 100,u(i,j) = 0;);
if(u(i,j) ge 800,u(i,j) = 1000;););
u('t','s') = 1000000;

gammas(i,j,ss) = uniform(0,1);
loop((i,j,ss),
if(gammas(i,j,ss) le 0.9999,gammas(i,j,ss) = 0;);
if(gammas(i,j,ss) gt 0.9999,gammas(i,j,ss) = 1;);  );

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
x(i,j);

Variable
z;

*Model pro maximalni tok v residualni siti ponicene utocnikem.
Equations obj     ucelova funkce pro uzivatele
          bal(n)  omezeni zachovavajici tok v uzlu
          src     omezeni pro tok z pocatecni uzel
          ter     omezeni pro tok do ciloveho uzlu
          xc(i,j)  kapacitni omezeni;

obj.. z =e= x('t','s');
bal(n).. sum(j, x(n,j)) - sum(i, x(i,n)) =l= 0;
src.. sum(n,x('s',n)) - sum(n,x(n,'s')) - x('t','s')  =l= 0;
ter.. sum(n,x('t',n)) - sum(n,x(n,'t')) + x('t','s')  =l= 0;
xc(i,j).. x(i,j) =l= u(i,j)*(1-gamma(i,j));

*=== EXPECTED VALUE===
model EV / obj, bal, src, ter, xc /;
gamma(i,j) = sum(ss, p(ss)*gammas(i,j,ss));
loop(ss,
solve EV max z using mip;
xs(i,j,ss) = x.l(i,j);
zs(ss) = z.l;);
zev = sum(ss, p(ss) * zs(ss));
display x.l, zev;
```

MM přístup k napadení sítě

```
option optcr=0.0001;
option ResLim = 100000000
option IterLim = 1000000000;
*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1 * n30, t/
n(i) /n1 * n30/
```

```
ss / 1*10 /;

Alias (i,j);
Parameter u(i,j)
          uu(i,j,ss)
          gammas(i,j,ss);

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
u(i,j) = uniform(0,1000);
loop((i,j),
if(u(i,j) le 100,u(i,j) = 0;);
if(u(i,j) ge 800,u(i,j) = 1000;););
u('t','s') = 1000000;

gammas(i,j,ss) = uniform(0,1);
loop((i,j,ss),
if(gammas(i,j,ss) le 0.9999,gammas(i,j,ss) = 0;);
if(gammas(i,j,ss) gt 0.9999,gammas(i,j,ss) = 1;););

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
xs(i,j,ss);

Variable
z;

*Model pro maximalni tok v residualni siti ponicene utocnikem.
Equations obj(ss)     ucelova funkce pro uzivatele
          bal(n,ss)   omezeni zachovavajici tok v uzlu
          src(ss)     omezeni pro tok z pocatecni uzel
          ter(ss)     omezeni pro tok do ciloveho uzlu
          xc(i,j,ss)  kapacitni omezeni
          neanticip(i,j,ss) neanticipativni omezeni;

obj(ss).. z =l= xs('t','s',ss);
bal(n,ss).. sum(j, xs(n,j,ss)) - sum(i, xs(i,n,ss)) =l= 0;
src(ss).. sum(n,xs('s',n,ss)) - sum(n,xs(n,'s',ss)) - xs('t','s',ss)  =l= 0;
ter(ss).. sum(n,xs('t',n,ss)) - sum(n,xs(n,'t',ss)) + xs('t','s',ss)  =l= 0;
xc(i,j,ss).. xs(i,j,ss) =l= u(i,j)*(1-gammas(i,j,ss));
neanticip(i,j,ss).. xs(i,j,'1') =e= xs(i,j,ss);

*=== MIN-MAX ===
model MM / obj, bal, src, ter, xc, neanticip /;
solve MM max z using mip;
display gammas, xs.l, z.l;
```

Dvoustupňové napadení sítě s kompenzací

```
option optcr=0.0001;
option ResLim = 100000000
option IterLim = 1000000000;
*Zavedeni mnoziny vsech uzlu a podmnoziny uzlu bez zdrojoveho a ciloveho uzlu.
Sets
i /s, n1 * n30, t/
n(i) /n1 * n30/
ss / 1*10 /;

Alias (i,j);

Parameter d(i,j)
          u(i,j)
          uu(i,j,ss)
          gammas(i,j,ss)
          p(ss);

p(ss) = 1/card(ss);

*Kapacitni matice. Pokud je hodnota = 0, hrana neexistuje.
u(i,j) = uniform(0,1000);
loop((i,j),
if(u(i,j) le 200,u(i,j) = 0;);
```

```
if(u(i,j) ge 800,u(i,j) = 1000;);
u('t','s') = 1000000;

d(i,j) = uniform(0,200);
loop((i,j),
if(u(i,j) le 10,u(i,j) = 0;);
if(u(i,j) ge 180,u(i,j) = 200););

gammas(i,j,ss) = uniform(0,1);
loop((i,j,ss),
if(gammas(i,j,ss) le 0.8,gammas(i,j,ss) = 0;);
if(gammas(i,j,ss) gt 0.8,gammas(i,j,ss) = 1;););

loop((i,j,ss),
if((u(i,j) eq 0) or (u(i,j) eq 1000), gammas(i,j,ss) = 0); );
uu(i,j,ss) =  u(i,j)*(1-gammas(i,j,ss));

*X(i,j) je promenna vyjadrujici tok na hrane (i,j).
Positive variable
xs(i,j,ss);

Binary variable
phi(i,j,ss);

Variable
z;

*Model pro maximalni tok v residualni siti ponicene utocnikem.
Equations obj         ucelova funkce pro uzivatele
          bal(n,ss)   omezeni zachovavajici tok v uzlu
          src(ss)     omezeni pro tok z pocatecni uzel
          ter(ss)     omezeni pro tok do ciloveho uzlu
          xc(i,j,ss)  kapacitni omezeni
          neanticip(i,j,ss) neanticipativni omezeni;

obj.. z =e= sum(ss, p(ss)*((xs('t','s',ss) - sum((i,j), d(i,j)*phi(i,j,ss)))));
bal(n,ss).. sum(j, xs(n,j,ss)) - sum(i, xs(i,n,ss)) =l= 0;
src(ss).. sum(n,xs('s',n,ss)) - sum(n,xs(n,'s',ss)) - xs('t','s',ss)  =l= 0;
ter(ss).. sum(n,xs('t',n,ss)) - sum(n,xs(n,'t',ss)) + xs('t','s',ss)  =l= 0;
xc(i,j,ss).. xs(i,j,ss) =l= u(i,j)*(1-gammas(i,j,ss)) + u(i,j)*phi(i,j,ss);
neanticip(i,j,ss).. xs(i,j,'1') =e= xs(i,j,ss);

*=== Twostage ===
model Twostage / obj, bal, src, ter, xc, neanticip /;
loop((i,j,ss),
if (uu(i,j,ss) GT 0,
         phi.up(i,j,ss) = 0;
         phi.lo(i,j,ss) = 0;);

if (uu(i,j,ss) EQ 0,
         phi.up(i,j,ss) = 1;
         phi.lo(i,j,ss) = 0;););

solve Twostage max z using mip;
display gammas, u, uu, d, phi.l, xs.l, z.l;
```